Robin Norman

ZX 81
Programación en BASIC



editores técnicos asociados, s. a.
BARCELONA

ZX 81

PROGRAMACION EN BASIC

ROBIN NORMAN

ZX 81

Programación en Basic

editores técnicos asociados, s. a.
Maignón, 26 - 08024 Barcelona - España
1984

Primera edición española, traducida del inglés por
FRANCISCO OLIVAN PEÑA
Licenciado en Ciencias Económicas

Título de la obra original:
ZX81 Basic Book
de Robin Norman
Newnes Technical Books

La edición original en lengua inglesa ha sido publicada por
© BUTTERWORTH & CO. (Publishers) LTD. 1982
Borough Green, Sevenoaks, Kent TN 15 8PH, England

© EDITORES TECNICOS ASOCIADOS, S. A. - Barcelona 1984
Depósito Legal: B-29580 - 1984
ISBN 84-7146-251-6

Impreso en España
Printed in Spain

GERSA, Industria Gráfica - Tambor del Bruc, 6 - Sant Joan Despí (Barcelona)

Indice de materias

Prefacio

Para un simple autor, resulta difícil mantenerse al día en el tema de los microordenadores. En 1985 escribí un libro dirigido a los propietarios del Sinclair ZX80 y me encontré con que a su publicación ya se anunciaba un nuevo producto Sinclair: el ZX81. El ZX80 era una máquina increíble, pero le dejaba a uno pensando: "Si hiciese ésto o esto otro...". Ahora, casi todas esas insuficiencias han sido cubiertas por el ZX81, que puede hacer casi todo y además por menos dinero.

La acogida dispensada a nuestro primer libro fue lo suficientemente buena como para animarnos a escribir otro sobre el ZX81. Dado que el BASIC del ZX81 incorporaba muchas características nuevas, resultaba evidente que la mayor parte del libro tendría que ser escrita de nuevo. No obstante, hemos utilizado el mismo esquema y nos hemos basado en las tres mismas hipótesis acerca de los lectores o lectoras del libro:

1 Es un recién llegado a la programación de ordenadores (si tiene alguna experiencia podrá saltarse algunos de los capítulos iniciales).

2 Tiene conectado ante sí un microordenador concreto: el Sinclair ZX81.

3 Desea aprender a utilizar todas las instrucciones del BASIC ZX81 a través de un curso estructurado y de intensidad creciente.

Usted no puede "leer" un libro como éste. Todo lo que obtenga de él será el resultado de la interacción de tres factores: Usted, este libro y el ZX81. Si en alguna ocasión se sorprende pensando: "¿Qué pasaría si...?¡Por lo que más quiera, pruébelo!. Desde luego no estropeará el ZX81 y seguramente aprenderá algo.

Ahora quiero expresar mi agradecimiento en primer lugar a Betty Clare, por su perfecto mecanografiado de mi difícil manuscrito. A Peter Chapman por sus útiles ideas y a mi familia, por su paciencia. Finalmente, a Clive Sinclair por prestarme el hardware otra vez. Es una máquina pequeña y maravillosa, con una, hay que admitirlo, memoria reducida (para eso está la expansión a 16K RAM), perfecta para principiantes y capaz de hacer un buen papel ante máquinas mucho más caras.

¡Deseo a todos que lo pasen bien programando!.

R.N.

Lista de programas

1

¿Qué hacen los ordenadores?

Este es un capítulo esencialmente filosófico. Si usted está impaciente por empezar, puede saltárselo ahora y leerlo más tarde.

Máquinas que controlan otras máquinas

En muchos aspectos el hombre es incapaz de competir con otros habitantes del planeta. Si ha llegado a dominar la Tierra ha sido gracias a su cerebro y al modo en que lo ha utilizado, al idear herramientas que no sólo le ahorran trabajo sino que le permiten hacer cosas que sin ellas serían inconcebibles.

En algún momento de la Prehistoria, el hombre comenzó a utilizar su cuchillo de piedra para hacer muescas en un palo. Desde entonces dispuso de dos tipos de herramientas:

(1) Aquellas que ayudan a sus músculos, haciendo trabajo mecánico.

(2) Aquellas otras que ayudan a su cerebro, calculando y memorizando.

Afortunadamente el hombre supo combinar ambos tipos de instrumentos, empleando las herramientas de cálculo para controlar las meramente mecánicas, haciendo posi-

bles sus mayores progresos. Esto ha venido sucediendo a lo largo de un dilatado período; los ejemplos siguientes lo demuestran:

Galgas sencillas para verificar que las flechas tenían la longitud y el diámetro requeridos.

Tornos que permiten la ejecución de complicados trabajos sobre metales por personas relativamente inexpertas.

Robots controlados electrónicamente para el montaje de automóviles en serie con un mínimo de intervención humana.

Por supuesto ha sido la Electrónica con su conocido "chip de silicona" la que ha extendido tan ampliamente el control automático de máquinas. Además nos ha traído algunos problemas sociales que tendrán que ser resueltos, pero esto es ya otra historia.

Máquinas de calcular 'puras'

Dado que este grupo incluye las máquinas de jugar a marcianitos, me ha parecido necesario poner la palabra "puras" entre comillas. Al hablar de ellas me refiero a aquellas máquinas de calcular que no se utilizan para controlar directamente otras máquinas. Hace ya tiempo que las venimos utilizando para auxiliarnos en Contabilidad, en investigación, e incluso para divertirnos. Aquí, el chip de silicona ha supuesto también una verdadera revolución, al posibilitar la reducción de tamaños y precios hasta el punto que podemos comprar una calculadora de bolsillo con el dinero que llevamos habitualmente encima o recibir un microordenador como regalo de cumpleaños.

Especializados y no especializados

Hay que distinguir dos tipos diferentes de calculadoras electrónicas:

(1) Aquellas que están especializadas, es decir, que fueron construidas con un conjunto de instrucciones en su interior para que realicen un trabajo concreto, como jugar al ajedrez o al mastermind, por ejemplo.

(2) Las que no están especializadas, es decir, las que están preparadas para recibir de nosotros instrucciones para hacer toda clase de trabajos, incluso jugar al ajedrez o al mastermind (por ejemplo el ZX81).

No obstante, incluso estos últimos poseen algún tipo de especialización. Sin ir más lejos, el ZX81 dispone en su interior de un juego de instrucciones para que comprenda un determinado lenguaje de programación: el BASIC.

Hardware y Software

En el mundillo de los ordenadores se utiliza un argot muy amplio, que facilita la comunicación entre los iniciados pero que constituye una barrera para el resto de personas. Nosotros usaremos el mínimo indispensable y además hemos incluido un glosario al final del libro.

Por "Hardware" entendemos todas las partes físicas del ordenador: el ZX81, el televisor, la impresora, el cassette, etc.

Llamamos "Software" a los programas e instrucciones que hacen que el ordenador pueda trabajar: el manual de instrucciones del ZX81, los programas permanentes que contiene y los programas que usted escriba.

Hemos avanzado desde lo general a lo particular. Vamos a ver ahora cómo podemos comunicarnos con el ordenador.

2

Hablemos con los ordenadores

El lenguaje del ordenador

Los humanos tenemos diez dedos. Por eso nos hemos acostumbrado a contar utilizando un sistema decimal basado en los dígitos que van de 0 a 9. Sin embargo, los ordenadores trabajan con números binarios. Un dígito binario (en adelante lo llamaremos 'bit') únicamente puede tener los valores 0 o 1, y eso es todo lo que el ordenador es capaz de llegar a distinguir.

Podemos escribir programas utilizando números binarios (en los primeros ordenadores había que hacerlo así), pero para los humanos los números binarios son difíciles de reconocer y manejar. Un método más práctico es escribir los programas en un lenguaje de bajo nivel, utilizando un código-máquina en números hexadecimales (en base 16), que son fácilmente convertibles en binarios. Los programas en código máquina son rápidos de ejecutar y económicos en la ocupación de memoria, pero no son adecuados para principiantes. La mayoría de usuarios de un ordenador se comunican con él en un lenguaje de alto nivel, basado en los números decimales y en un conjunto de palabras fácilmente reconocibles. He aquí algunos de los lenguajes de alto nivel más comunes:

FORTRAN FORmula TRANslation. Ideado para usarlo en Ciencias e Ingeniería.

COBOL COmmercial Business Oriented Language. Para aplicaciones comerciales y administrativas principalmente.

BASIC Beginners All-purpose Symbolic Instruction Code. Concebido para principiantes y usos generales.

De vez en cuando aparece un lenguaje nuevo que pretende tener algunas ventajas. Hoy por hoy, el BASIC es el lenguaje más ampliamente usado en la actual generación de microordenadores.

El ordenador por sí solo es incapaz de comprender estos lenguajes de alto nivel, por lo que se le dota de programas para traducirlos a números binarios vía código-máquina.

La memoria del ordenador

La memoria de un ordenador es algo así como un panel con un gran número de celdillas, cada una de las cuales contiene un número binario compuesto de 8 bits (a este número en adelante le llamaremos byte). La capacidad de la memoria se expresa en unas unidades llamadas "K". Un K es igual a 1024 bytes. En un ordenador como el ZX81 hay dos tipos de memoria:

Memoria de sólo lectura (ROM)

Contiene el programa necesario para hacer trabajar al ordenador y para traducir las instrucciones BASIC a un código binario. La ROM es permanente y no se borra cuando se desconecta el ordenador. El ZX81 dispone de una ROM de 8K.

Memoria de acceso aleatorio (RAM)

Guarda todos los datos y programas que usted introduzca. No es permanente, por lo que si usted desconec-

ta el ZX81, aunque sólo sea un instante, el contenido de la RAM se borra. El ZX81 tiene una RAM de 1K, que puede convertirse en 16K aplicándole un accesorio de ampliación de memoria.

Entradas y salidas

El ZX81 tiene un teclado de estructura similar al de una máquina de escribir para que podamos introducir datos e instrucciones. Para ver qué es lo que estamos introduciendo y para que el ZX81 nos muestre el resultado de lo que hace, utilizamos un televisor ordinario sintonizado en UHF.

Si queremos imprimir un programa o un resultado, emplearemos una impresora. El ZX81 utiliza un juego de caracteres no standard, por lo que la única impresora que puede usarse es la suya propia.

Almacenamiento permanente

Hemos visto que si se desconecta el ZX81 el contenido de la RAM se borra, perdiéndose datos y programas que quizá nos costó horas escribir. Aún en el caso de que los tengamos impresos, nos veremos obligados a teclearlos de nuevo. Necesitamos una memoria permanente para guardar programas y datos todo el tiempo que queramos. Para esto el ZX81 utiliza un cassette corriente, con el que programas y datos se graban en una cinta y pueden guardarse indefinidamente, bastando volver a cargarlos en el ZX81 cuando deseemos emplearlos de nuevo.

Echemos una ojeada al futuro

Vamos a mirar en la bola de cristal para hacer una lista de características que desearíamos que tuvieran los ordenadores personales en el futuro:

(1) Mayor estandarización, de tal modo que los programas sean más intercambiables y los ordenadores capaces de comunicarse entre sí.

(2) Comunicación con el ordenador de viva voz, tanto para la entrada como para la salida.

(3) Salida impresa barata, preferiblemente en forma de máquina de escribir que actúe también como impresora.

(4) Memoria barata, ilimitada y permanente.

(5) Salida por pantalla en color y de alta definición, de calidad similar a la de la TV.

(6) Una amplia gama de software para la empresa, el hogar, la enseñanza y el ocio.

(7) Posibilidad de conectar con ordenadores grandes a través de las redes telefónica o de TV, para acceder a una información virtualmente ilimitada sobre cualquier tema.

3

Programación en BASIC

El BASIC es uno de los lenguajes de alto nivel más ampliamente utilizados, en especial por la actual generación de microordenadores. Existen diferentes versiones del BASIC, de la misma manera que hay distintos dialectos del castellano. ¡Pero no hay que desanimarse!. En todas las versiones del BASIC se reconoce fácilmente el tronco común que las originó (el BASIC fue desarrollado en el Dartmouth College, New Hampshire, USA) y cuando se ha aprendido una de ellas se puede pasar fácilmente a otra en un ordenador distinto. El BASIC en 8K ROM del Sinclair ZX81 es una versión bastante completa, que tiene pocas características no standard y que por muchas razones es excelente para los principiantes.

El primer programa

Vamos a emplear un ejemplo simpático para hacer nuestros primeros pinitos en programación. ¿Recuerdan el episodio del Aprendiz de Brujo en la película Fantasía, de Walt Disney?. Es uno de mis favoritos. En él Mickey Mouse es el aprendiz de un brujo que le ha encargado un trabajo aburrido: Llenar un gran tanque con agua de la fuente. Como Mickey es un chico listo, decide programar una escoba para que haga su trabajo mientras él duerme la siesta.

Cuando se escribe un programa, el orden de las instrucciones es muy importante; por eso se las numera. El primer intento de Mickey podría haber sido así:

1 Coge un cubo y ve a la fuente.

2 Llena el cubo con agua.

3 Lleva el cubo al tanque.

4 Vacía el cubo en el tanque.

Hasta ahora todo va bien. La escoba ha vaciado un cubo de agua en el tanque. Mickey podría repetir las mismas instrucciones una y otra vez, numerándolas con 5, 6, 7, 8, 9, 10, 11, 12, etc. Pero no. Ha leido el capítulo 12 del libro mágico y todo lo que hace es añadir una sola instrucción:

5 GO TO 1 (ve a la instrucción 1)

Con esto ha creado un bucle. La escoba sigue el programa con toda exactitud y va y viene alegremente llenando, transportando y vaciando cubos. Mientras tanto Mickey duerme ...

* * * * *

... hasta que al cabo de un rato se despierta sobresaltado con el agua hasta las rodillas. Es fácil adivinar lo ocurrido: ¡Olvidó decirle a la escoba cuando tenía que parar!. Mickey, preso del pánico, rompe la escoba en dieciseis trozos, pero cada uno de ellos se levanta y se pone a transportar cubos. Por suerte el mago regresa justo a tiempo. El conoce bien el arte de programar escobas y sabe que todo bucle debe contener un test de salida; de lo contrario el bucle se ejecutaría una y otra vez indefinidamente. Este paso tan importante siempre contiene la palabra 'IF' y nosotros lo conocemos como salto condicional.

El programa definitivo, con su instrucción IF y numeración nueva queda así:

1 Coge un cubo y ve a la fuente.

2 Llena el cubo con agua.

3 Lleva el cubo al tanque.

4 Vacía el cubo en el tanque.

5 Si (IF) el tanque está lleno, vete a 1 (GO TO 1).

6 Avisa que el tanque está lleno.

7 Para.

La instrucción IF tiene que estar situada dentro del bucle, de tal manera que cada vez que la escoba lo ejecute compruebe si el tanque está o no lleno y actúe en consecuencia.

Bueno, todo esto no es gran cosa, pero nos ha servido para señalar cuatro puntos que cobrarán toda su importancia cuando empecemos a programar de verdad:

(1) Un programa BASIC está formado por una serie de instrucciones.

(2) Todas las instrucciones están numeradas, por lo que el ordenador puede ejecutarlas en el orden establecido.

(3) Se puede hacer que el ordenador ejecute una parte del programa muchas veces utilizando una instrucción GO TO. A esto lo llamamos bucle.

(4) Un bucle debe contener una instrucción denominada salto condicional, que detiene el ordenador o lo envía fuera del bucle cuando se cumple determinada condición. La palabra mágica que identifica esta instrucción es IF.

4

El Hardware

Antes de conectar el ordenador, conviene que lea usted el capítulo 1 del manual de instrucciones. Las tres partes principales del equipo son el ZX81 propiamente dicho, la unidad de alimentación y un televisor sintonizado en UHF (mejor si es en blanco y negro). El ZX81 puede trabajar perfectamente sin pantalla, pero usted necesita comunicarse con él. Monte las tres unidades tal como se describe en el manual. Conecte a la red eléctrica y entonces gire el botón de sintonía del televisor hasta que en la pantalla aparezca la letra K en un cuadrado negro; a esto le llamaremos cursor [K].

El teclado del ZX81

Merece la pena esforzarse un poco para dominar el teclado del ZX81. Ha sido diseñado cuidadosamente para que cada tecla haga el máximo número de cosas posible. Mediante el teclado puede hacerse aparecer en la pantalla lo siguiente:

(1) El conjunto de palabras clave

Se trata de las palabras que aparecen impresas en blanco encima de las teclas que corresponden a las letras. Son instrucciones que indican al ZX81 lo que de-

be hacer. Cuando el cursor [K] está en pantalla, si presionamos una tecla de letra aparecerá la palabra clave que corresponda. Hagamos la prueba: Presionamos la tecla Y e inmediatamente aparece la palabra **RETURN** y el cursor cambia de [K] a [L]. Probablemente usted se estará preguntando cómo se puede elegir entre la letra o la palabra clave correspondientes a la tecla que se pulsa. La respuesta es simple: ¡Usted no puede!. Es el ZX81 el que sabe cuándo se precisa una palabra clave o una letra, hace la elección correcta y la indica en pantalla mediante el cursor apropiado.

Para facilitar su localización en el teclado, muchas palabras clave han sido situadas sobre su letra inicial o cerca de ella. En este libro, igual que en el manual de instrucciones, todas aquellas palabras que se obtienen con una sola pulsación de una letra están impresas en negritas. Por ejemplo:

PRINT **ABS** **TO**

(2) El alfabeto

Usted puede escribir en la pantalla cualquier letra (o espacio, o punto) pulsando la tecla correspondiente mientras vea el cursor [L]. Pruébelo y observe que el cursor se va moviendo por delante de lo que usted ha escrito. El próximo carácter que teclee aparecerá exactamente en la posición que ahora ocupa el cursor.

(3) El conjunto de números

Pueden teclearse igual que las letras, con la salvedad de que el cursor en pantalla puede ser [K] o también [L]. No cambian el cursor de [K] a [L]. Procure no confundir el cero, Ø, con la letra O. Cuide también las confusiones entre el número 1 y la letra I.

(4) Un conjunto de caracteres diversos

Si mantiene pulsada la tecla **SHIFT** (en la parte in-

ferior izquierda del teclado) y toca casi cualquier tecla de letra o número, puede imprimir una de una colección de palabras, signos de puntuación y símbolos matemáticos, que figuran en rojo en la parte superior de cada tecla. Las excepciones son **SHIFT** 1 y **SHIFT** 5 a Ø, que no imprimen nada en la pantalla. Luego veremos para qué sirven.

Cinco de estas palabras (**STOP**, **LPRINT**, **SLOW**, **FAST**, **LLIST**) son palabras clave que hay que usar con el cursor [K]. El resto se emplean solo con el cursor [L].

(5) Un conjunto de funciones

Aparecen impresas en blanco bajo cada tecla de letra. Mantenga pulsada **SHIFT** con el cursor [L] visible y oprima la tecla **FUNCTION/NEWLINE,** situada a la derecha. Verá que el cursor [L] cambia a [F]. Si ahora pulsa una tecla de letra (excepto V), aparecerá la función correspondiente en pantalla. Las funciones suelen utilizarse de una en una; por tanto, el cursor vuelve automáticamente a [L].

(6) Un conjunto de caracteres gráficos

Pulse la tecla **GRAPHICS** (**SHIFT** 9) con [L] en pantalla y verá cómo éste cambia a [G] (que permanecerá hasta que toque **GRAPHICS** otra vez). Ahora puede experimentar toda una serie de efectos especiales.

Toque cualquier tecla de letra o número y obtendrá su impresión inversa (es decir, blanco sobre negro), útil para dar énfasis. Mantenga presionada **SHIFT** y pulse una de las 20 teclas de la izquierda; en pantalla aparecerá uno de los signos gráficos negros, blancos y grises, que usaremos luego para dibujar y hacer gráficos. La tecla **SHIFT** con cualquier otra permite obtener la imagen inversa del carácter superior correspondiente.

Sabemos ya cómo obtener en pantalla cualquier carácter; por tanto podemos proseguir con la programación. Lo primero que debemos hacer es borrar la panta-

lla. Para esto lo más rápido es pulsar **EDIT** (**SHIFT** 1), que limpia la pantalla y deja el cursor [K]. Igual podemos hacerlo desenchufando un momento, pero recuerde que así se borran también todos los datos y programas introducidos en el ZX81. ¡Es el último recurso!.

En este capítulo hemos aprendido ...

A instalar el ZX81

Todos los conjuntos de caracteres a que podemos acceder desde el teclado.

5

El primer programa

Borremos los programas anteriores

Antes de introducir un nuevo programa en el ZX81 hay que borrar los anteriores. Bueno, ya sabemos que aún no hay nada, pero no importa, practiquemos un poco. Al acabar el capítulo anterior dejamos el cursor en posición [K]. En este caso todo lo que tenemos que hacer es presionar la tecla A y en la pantalla aparecerá la palabra clave **NEW**. Ahora, para que surta efecto, tenemos que introducirla en el ZX81 pulsando **NEWLINE**, a la derecha del teclado. Al hacerlo, **NEW** desaparece de la pantalla y reaparece [K]. El ZX81 ha obedecido el comando: 'borra cualquier programa anterior y prepárate para recibir uno nuevo'.

Comandos e instrucciones

Humpty Dumpty decía: 'cuando utilizo una palabra su significado es exactamente aquel que yo elijo'. Acabamos de utilizar las instrucciones **NEW** y **NEWLINE**; en lo sucesivo las llamaremos comandos. Los comandos no forman parte de un programa. Son órdenes que se obedecen una vez y luego se olvidan. Casi todas las palabras clave y **EDIT**, **LPRINT**, **SLOW**, **FAST** y **LLIST** son aceptadas por el ZX81 como comandos. **INPUT** da error. **FOR**, **NEXT**, **PAUSE** y **SCROLL** no son útiles.

Por otra parte, las sentencias o instrucciones propiamente dichas forman parte de las líneas numeradas de los programas. El ZX81 las memoriza y obedece cada vez que se ejecuta el programa. Salvo **CONT**, cualquier palabra clave puede constituir una instrucción, así como **STOP**, **LPRINT**, **SLOW**, **FAST** y **LLIST**.

Escribamos un programa

¡Ya era hora!. Ya vimos en el capítulo 3 que todas las líneas de un programa **BASIC** tienen que estar numeradas; por tanto, teclee un número de línea, como "1Ø" por ejemplo. El cursor sigue siendo [K] dado que lo que sigue a un número de línea debe ser una palabra clave. Ahora pulse "P". En pantalla tenemos:

1Ø **PRINT**[L]

Continúe tecleando:

1Ø **PRINT** "REGLA PRIMERA EN BASIC[L]

Introduzca todo esto en la memoria del ZX81 pulsando **NEWLINE**. ¿Qué ha pasado?. Pues que ha aparecido otro cursor: el [S]. Esto significa que hemos cometido un error de sintaxis, algo equivocado en la línea que impide que sea admitida. Inmediatamente recordamos que las comillas siempre van en parejas, las cerramos y tocamos de nuevo **NEWLINE**. Esta vez la línea 1Ø salta a la parte superior de la pantalla y el cursor [K] reaparece al pie. La línea ha sido aceptada y ya podemos escribir otra.

Pero por el momento no lo vamos a hacer. Vamos a ejecutar el programa tal como está. Pulse "R" para situar en pantalla el comando **RUN** y seguidamente toque **NEWLINE**.

Cuando haya pasado el primer momento de euforia observe cuidadosamente lo que ha ocurrido. Todas las palabras y espacios entrecomillados han sido impresos tal como estaban, mientras que el número de línea, cursores y comillas han desaparecido (servían sólo pa-

ra indicar al ZX81 qué debía hacer). Lo que hemos hecho con la línea 1Ø ha sido imprimir en pantalla una cadena de caracteres. Las cadenas de caracteres van entrecomilladas y pueden contener cualquier cosa procedente del teclado excepto las comillas **SHIFT** P (si desea que aparezcan comillas use **SHIFT** Q). En la parte inferior de la pantalla vemos Ø/1Ø, que es un código mediante el cual el ZX81 nos informa que "el programa se ha ejecutado sin problemas y ha terminado en la línea 1Ø". El programa permanece en memoria hasta que usted lo borre, teclee **NEW** o desconecte. Si lo desea puede verlo de nuevo pulsando **NEWLINE**.

Oculto por el código de información se encuentra el cursor [K], listo para iniciar la próxima línea de programa. Pasemos al capítulo siguiente y escribamos algunas más.

En este capítulo hemos aprendido ...

Comandos

NEW para borrar programas anteriores.

NEWLINE para introducir comandos y líneas de programa.

RUN para ejecutar los programas.

Instrucciones

PRINT para imprimir en pantalla cadenas de caracteres.

Otros

Comandos e instrucciones

Cursores

Errores de sintaxis

Códigos de información

6

¡Programe bien!

La segunda línea de nuestro programa

Para que el programa introducido aparezca en pantalla pulse **NEWLINE**. Luego teclee la segunda línea así:

```
20  PRINT "TODA LINEA NECESITA UN NUMEROS"[L]
```

¡Perdón!. Hemos cometido un error, pero no hay que preocuparse. Mantenga presionada **SHIFT** y toque **RUBOUT** dos veces. Verá que el cursor [L] retrocede dos posiciones, borrando las comillas y la S. Teclee otra vez las comillas y presione **NEWLINE**, con lo que la línea 2Ø se añadirá al programa. Ejecútelo y obtendrá en pantalla la siguiente salida:

```
REGLA PRIMERA EN BASIC
TODA LINEA NECESITA UN NUMERO
```

además del código Ø/2Ø.

Hasta ahora he venido indicando el tipo y posición del cursor al introducir líneas de programa. En lo sucesivo lo omitiré, salvo en aquellos casos en que haya una razón que aconseje lo contrario.

Mejoremos los programas

Obtendremos un resultado más pulcro si entre las dos líneas de la salida dejamos un espacio en blanco. Pruébelo. Teclee:

15 **PRINT**

e introdúzcalo pulsando **NEWLINE**. ¿Qué demonios estamos tratando de hacer?. ¿Imprimir qué?. Bueno, ejecute el programa y lo verá. Funciona perfectamente, ¡por supuesto!. Cuando el ZX81 se encuentra con una instrucción **PRINT** imprime lo que ésta le diga. La línea 15 le dice que no imprima nada, ¡y nada resulta impreso!.

Finalmente añadiremos un comentario que nos diga algo sobre el programa. Teclee:

5 **REM**** MI PRIMER PROGRAMA

La instrucción **REM** equivale a decir: 'Ignora el resto de esta línea. Es sólo un recordatorio para el programador'. Los asteriscos ** se han añadido simplemente para que la línea **REM** resalte mejor.

Numeración y listado

A la mayoría de ordenadores hay que pedirles expresamente que listen el programa, pero no al ZX81 que lo hace tan pronto como usted toque **NEWLINE** o añada otra línea. Con toda seguridad usted habrá notado que el ZX81 ordena numéricamente las líneas, aunque hayamos tecleado 10, 20, 15, 5. Imagino que también habrá adivinado que elegimos números de línea no consecutivos para poder realizar inserciones fácilmente, como hicimos con las líneas 15 y 5. Al ZX81 no le interesa el valor de estos números, sino sólo su orden. Podemos elegir entre 1 y 9999. ¡Seguro que son suficientes!.

Borrar líneas completas

Supongamos que queremos borrar la línea 5 para

ahorrar memoria (con frecuencia éste es el triste destino de las líneas **REM**. Basta con teclear el número de línea 5 y pulsar **NEWLINE**. Puede hacerlo tantas veces como desee. El ZX81 borrará siempre la línea antigua y la sustituirá por la nueva.

Hagamos ahora un par de ejercicios para practicar lo aprendido en los dos últimos capítulos.

Ejercicio 6.1 Cambio de línea

Borre las líneas 15 y 2Ø y cambie la 1Ø de tal modo que se imprima el mensaje:

TRES LINEAS SE VAN, UNA SE QUEDA

Ejercicio 6.2 Su dirección

Borre el programa anterior con una sola palabra clave (¿recuerda cuál?). Escriba un programa que imprima su nombre y dirección como lo haría en un sobre. Deje intervalos de 1ØØØ entre los números de línea, sólo para comprobar que es posible.

En este capítulo hemos aprendido ...

Instrucciones

PRINT para dejar una línea en blanco.

REM para comentarios que el ZX81 ignora.

Otros

RUBOUT para borrar caracteres de uno en uno en la línea que se está escribiendo.

Listado automático del programa al pulsar **NEWLINE**.

Números de línea no consecutivos para facilitar inserciones ulteriores.

Borrar y cambiar líneas.

7

¿Sumar? ¡Ningún problema!

Hasta ahora he venido recordando que hay que presionar **NEWLINE** para introducir en el ZX81 comandos y líneas de programa de la parte inferior de la pantalla. ¡En lo sucesivo tendrá que recordarlo usted!.

Palabras clave en funciones de comando

Hemos utilizado la palabra **PRINT** como instrucción en líneas de programa, pero ya vimos en el capítulo 5 que muchas palabras clave pueden utilizarse también como comandos. Haga la prueba, teclee:

PRINT "ESTO ES UN COMANDO DE SALIDA"

El ZX81 obedece una sola vez y luego la olvida. El código Ø/Ø muestra que no hay número de línea.

Números y expresiones

Podemos hacer que el ZX81 imprima números de la misma manera que imprime cadenas de caracteres. La única salvedad es que en este caso no hay que utilizar comillas. Pruebe:

PRINT 99   **PRINT** 3.74

PRINT Ø.Ø75 **PRINT** .625

El punto actúa como punto decimal. El cero a la izquierda del punto decimal puede omitirse.

Una expresión está integrada por números y operadores. Por ejemplo, 5+3. Si ordenamos al ZX81 que imprima una expresión, calculará e imprimirá el resultado. Teclee:

PRINT 5+3

La respuesta, 8, aparecerá en la parte superior de la pantalla. De esta manera hemos utilizado el ZX81 como si fuera una calculadora de bolsillo, pero con la ventaja de que antes de obtener el resultado podemos ver la expresión completa y corregirla si es necesario.

Operadores y prioridades

Todos recordamos los cuatro operadores aritméticos básicos. Los signos - y + se encuentran en **SHIFT** J y **SHIFT** K. En lugar de x o "multiplicado por" utilizamos * (**SHIFT** B) y en vez de : o "dividido por" usamos / (**SHIFT** V). Esto es igual en cualquier tipo de ordenador. Además tenemos ** (**SHIFT** H) que significa "elevado a la potencia":

2**3 = 2^3 = 2×2×2 = 8

Practique tecleando expresiones simples compuestas de dos números y un operador:

PRINT 25-17 **PRINT** 7-12 **PRINT** 9*11
PRINT 63/9 **PRINT** 125/48

Observe que el ZX81 opera satisfactoriamente con números negativos y con decimales. Si usted teclea:

PRINT 2/3 y **PRINT** 1/3

la respuesta constará de 8 decimales, con redondeo.

¿Qué sucede cuando las expresiones son más complejas, tal como la siguiente?:

2 + 3*4

Si calculásemos a mano el valor de una expresión como ésta seguiríamos un determinado orden en la ejecución de las operaciones. El ZX81 y la mayoría de ordenadores hacen lo mismo.

Mayor prioridad	**	Potenciación
↓	* y /	Producto y cociente
Menor prioridad	+ y -	Adición y sustracción

Por tanto la expresión anterior se calcula en dos etapas:

(1) 3*4 = 12
(2) 2 + 12 = 14 (resultado)

Pruebe con las expresiones que siguen y con otras que usted mismo puede idear:

```
PRINT 7*2 - 5        PRINT 9 - 12/6
PRINT 3**3 + 5       PRINT 38 - 5**2
```

Asegúrese de que las respuestas que obtiene son las correctas.

Paréntesis

Si queremos que el ZX81 modifique su prioridad en la ejecución de operaciones tendremos que utilizar paréntesis. El ZX81 respeta las reglas matemáticas y calcula en primer lugar las expresiones que están dentro de los paréntesis. Compare estos dos casos:

PRINT 2 + 3*4	**PRINT** (2 + 3)*4
(1) 3×4 = 12	(1) 2 + 3 = 5
(2) 2 + 12 = 14	(2) 5×4 = 20

Compruebe que el ZX81 da bien las respuestas. Usted puede utilizar cuantos paréntesis desee, siempre que no olvide cerrar ninguno, ya sean contiguos o separados. Cuando el ZX81 se encuentra con paréntesis múltiples empieza el cálculo por la parte interior y va avanzando hacia el exterior. No dude en utilizar más paréntesis que los estrictamente necesarios si con esto una expresión resulta más inteligible para usted.

Ejercicio 7.1

Calcule la respuesta de las expresiones siguientes y luego verifíquelas con el ZX81.

```
((7-5)*(30/12))**3
(((6*8)-(23-11))/(5+7))**2
```

Notación científica

Teclee estos comandos y observe atentamente las respuestas:

```
PRINT .ØØØØ7
PRINT 7/1Ø**5
PRINT 7ØØØØØØØØØØØØ (12 Øs)
PRINT 7*1Ø**12
```

y ahora éstos:

```
PRINT .ØØØØØ7
PRINT 7/1Ø**6
PRINT 7ØØØØØØØØØØØØØ (13 Øs)
PRINT 7*1Ø**13
```

Cuando maneja números muy grandes o muy pequeños, el ZX81 los imprime en notación científica:

7E+13 es lo mismo que 7*1Ø**13 o $7 \times 1Ø^{13}$

7E-6 es 7/1Ø**6 o $7/1Ø^{6}$ o $7 \times 1Ø^{-6}$

Este método es muy utilizado en calculadoras.

Si queremos se puede emplear la notación científica para representar los números que introducimos en el ZX81. Teclee:

```
PRINT 7E-5
PRINT 7E-6
```

Si dispone de espacio, el ZX81 pasará estos números a notación ordinaria.

En este capítulo hemos aprendido ...

Comandos

PRINT para imprimir cadenas de caracteres o números o resultados.

Otros

Operadores matemáticos y prioridades

Paréntesis para alterar prioridades

Notación científica

8

Variables

Hemos aprendido a utilizar comandos para que el ZX81 imprima en pantalla números o resultados. Esto mismo lo podemos conseguir desde el programa, pero no resulta especialmente útil dado que disponemos de una instrucción mucho más potente: **LET**.

Definición de variables con LET

Borre la memoria del ZX81 con **NEW** y teclee:

10 **LET** X = 5

Ejecútela. No aparecerá ninguna salida salvo el mensaje 0/10. ¿Qué hemos hecho esta vez?. En lenguaje ordinario diríamos que hemos etiquetado con una X una celdilla de memoria y hemos guardado un 5 en ella. En otras palabras, hemos definido la variable X y le hemos asignado el valor 5.

Con el contenido de X podemos hacer muchas cosas. Podemos imprimirlo:

20 **PRINT** X (y **RUN**)

Podemos emplearlo en expresiones:

```
30  PRINT 100 * X
40  PRINT X ** 3
```

Vea que aunque hemos utilizado el contenido de la "celdilla" X en las líneas 20, 30 y 40, el valor 5 sigue allí. Compruébelo:

```
50  PRINT X
```

El 5 inicial continúa estando allí, pero podemos cambiarlo si queremos:

```
60  LET X = 999
70  PRINT X
```

La línea 60 dice: "Vacía el contenido de la celda X y pon en ella el valor 999". Podemos modificar el valor de una variable tantas veces como queramos; por eso precisamente se llama variable.

Denominación de variables

El número de variables que podemos emplear en un programa está limitado únicamente por la memoria disponible, pero en cualquier caso todas ellas deben tener nombres diferentes. El ZX81 ofrece una amplia gama de posibilidades de elección de nombres de variable. Simplemente hay que seguir estas reglas:

(1) Los nombres de variable deben empezar por una letra, no por un número.

(2) Los nombres de variable pueden tener letras y cifras, pero no blancos (se ignoran) o cualquier otro signo (no se admiten).

Generalmente utilizamos nombres mnemotécnicos y cortos para economizar memoria: T para total, P para peso, etc. Practique con nombres de variable que se le ocurran a usted y vea qué pasa cuando no respeta las reglas anteriores.

Instrucciones LET más complejas

Nuestra instrucción tiene la forma general:

LET Nombre de variable = ...

¿Qué es lo que podemos poner a la derecha del signo igual?. Veamos unos ejemplos: El primero ya lo conocemos:

(1) Un número:
LET B = 75

(2) Una expresión compuesta por números:
LET C = 23*45

(3) Una expresión que use otras variables, con o sin números:
LET A = C **LET** V = B**3

¡Atención!. A la derecha del signo igual usted sólo puede utilizar variables que previamente hayan sido definidas, de lo contrario el ZX81 se las rechazará.

(4) Una expresión que utilice la misma variable que a la izquierda del signo =.

LET B = B+10 **LET** A = A*X

En Algebra nunca actuaríamos así. Recuerde que esto no son ecuaciones. Estamos diciendo cosas como: "Toma el valor de la celda B, súmale 1Ø y pon este nuevo valor en esa celda".

Cómo utilizar las variables

Si conocemos el radio (R) de un círculo, podemos emplear las ecuaciones siguientes para calcular el diámetro (D), circunferencia (C) y área (A).

$D = 2R$
$C = \pi D$
$A = \pi R^2$

Como valor de π tomaremos 3.14. Vamos a programar todo esto. Empezaremos por definir R, el radio del círculo en cm.

```
10  LET R = 5
```

Ahora definimos y calculamos las otras variables:

```
20  LET D = 2*R
30  LET C = 3.14*D
40  LET A = R**2*3.14
```

Finalmente imprimimos los resultados:

```
50  PRINT R
60  PRINT D
70  PRINT C
80  PRINT A
```

Ejecute el programa y verifique los resultados con una calculadora de bolsillo. Si deseamos que los resultados sean menos impersonales, podemos imprimir títulos:

```
45  PRINT  "RADIO CONOCIDO = "
55  PRINT  "DIAMETRO = "
65  PRINT  "CIRCUNFERENCIA = "
75  PRINT  "AREA = "
```

Usted puede cambiar el dato inicial, el radio, escribiendo de nuevo la línea 10.

Bueno, para empezar no ha sido un mal programa, pero el orden de la salida impresa deja bastante que desear. En el próximo capítulo lo mejoraremos un poco.

<u>Ejercicio 8.1</u> Cambio de moneda

Supongamos que en el mercado de cambios la libra esterlina cuesta 1.90 dólares. Escriba un programa que imprima el número de dólares que puede obtener por 75 libras y cuántas libras son necesarias para comprar 250 dólares.

Ejercicio 8.2 Paracaidismo

Un miembro del equipo Halcones salta desde su avión a 3.000 metros. La distancia de caída viene dada por la ecuación $S = AT^2/2$ donde A es la aceleración de la gravedad (9,8 m/s/s) y T es el tiempo en segundos desde el momento del salto (se desprecia la resistencia del aire). Escriba un programa que calcule su altura después de 10 segundos. En el supuesto que tenga que abrir el paracaídas a 500 metros sobre el suelo, utilice el programa para calcular aproximadamente cuántos segundos durará su caída libre.

En este capítulo hemos aprendido ...

Instrucciones

LET para definir variables.

PRINT para imprimir el valor actual de una variable.

Otros

Reglas para la denominación de variables.

Diversos empleos de las variables.

9

Signos de puntuación

Está claro que nos hemos excedido en el número de instrucciones **PRINT** que hemos utilizado. Podemos simplificar el aspecto que ofrece la pantalla aprovechando más cada línea. Véalo:

```
10  PRINT  "AREA DE UN CUADRADO"
20  PRINT
30  LET S = 4
40  PRINT  "LADO = ";S;" CM"
```

Ejecútelo y observe el resultado de la línea 40. Lo más importante son los puntos y coma, que equivalen a decir: "No saltes de línea. Imprime la siguiente información en las posiciones contiguas". Se puede utilizar el punto y coma separando diversas informaciones dentro de la misma línea **PRINT**, o bien al final de ella. En este último caso la próxima instrucción **PRINT** ocasionará una impresión contigua a la anterior. Vea que para tener un espacio en blanco entre S y CM, hemos incluido uno dentro de las comillas.

Ahora modifique el programa así:

```
35  LET A = S*S
40  PRINT  "LADO = ";S;" CM","AREA = ";A;" SQCM"
```

Hemos usado otro importante signo de puntuación: la

coma. Cada línea puede dividirse en dos mitades y la coma significa: "Imprime lo que sigue partiendo del principio de la siguiente mitad de línea". Si lo desea puede utilizar comas en grupos. Cada una de ellas desplaza la impresión al principio de la siguiente mitad de línea.

Respecto al punto, ya sabemos que cumple la función de punto decimal. El resto de signos de puntuación pueden utilizarse en cadenas de caracteres, pero no tienen una utilidad especial.

Tabulación

¿Cuántos caracteres puede contener una línea de pantalla?. Cuéntelos:

20 **PRINT** "01234567890 1234...

Cuando se agote la línea los números empezarán a salir en la siguiente, pero recuerde que 20 **PRINT** " también ocupa espacio. Deje de introducir cifras cuando el cursor [L] esté exactamente debajo del primer 0, cierre las comillas y pulse **NEWLINE**. Ahora, si ejecuta el programa obtendrá una línea completa de 32 cifras (para comprobarlo teclee otra vez con una cifra más). Podemos repetir la línea imprimiendo las decenas tras diez blancos:

10 **PRINT** "   111111111122222222223 3"

Ahora vea el efecto de la coma sobre la salida:

30 **PRINT**
40 **PRINT** "PRIMERA MITAD","SEGUNDA MITAD"

Ahora modifique y amplíe el programa así (recuerde que **TAB** forma parte del conjunto de funciones):

40 **PRINT** "UNO"
50 **PRINT TAB** 7;"DOS"
60 **PRINT TAB** 15;"ATO"
70 **PRINT TAB** 20;"MI ZAPATO"

Es evidente lo que ocurrirá: **TAB** n; mueve la salida impresa hasta la posición n. Es necesario poner un punto y coma después de **TAB** n (podría ponerse una coma, pero no es aconsejable).

Es frecuente tener que imprimir varias cosas en la misma línea. No hay problema. Sencillamente utilice el punto y coma para impedir que el ZX81 salte de línea. He aquí un ejemplo:

```
4Ø PRINT TAB 8; "EXTRACTO DE CUENTA"
5Ø PRINT
6Ø PRINT
7Ø PRINT "FECHA";TAB 8;"DEBE";TAB 14;"HABER";TAB
   24;"SALDO"
```

Podemos imprimir números, expresiones o variables en las posiciones indicadas por **TAB** igual que lo hemos hecho con cadenas de caracteres. Veamos algunas directrices para el uso de **TAB** que nos serán muy útiles más adelante:

(1) No es imprescindible poner un número después de **TAB**. Podemos utilizar una variable previamente definida o una expresión que contenga números y variables.

(2) Si el número que sigue a **TAB** es decimal será redondeado al entero más próximo (7.5 se redondeará a 8).

(3) Si el número que sigue a **TAB** es mayor que 31, será dividido por 32 y se tomará el resto.

<u>Ejercicio 9.1</u> Círculos

Vuelva al capítulo 8 y escriba de nuevo el último programa de tal modo que su salida en pantalla sea:

```
     DATOS IMPORTANTES DE UN CIRCULO
SI EL RADIO ES 5 CM
DIAM = 10 CM  CIRCUNF = 31.4 CM
        AREA = 78.5 SQCM
```

Una vez escrito el programa no lo borre, lo usaremos en el próximo capítulo.

En este capítulo hemos aprendido ...

; , y **TAB** para modificar las posiciones de impresión en pantalla.

10

¡Un error lo comete cualquiera!

Hemos visto ya dos procedimientos para corregir errores en un programa. Podemos utilizar **RUBOUT** si queremos borrar algo en la misma línea que estamos escribiendo. También podemos borrar o sustituir una línea anterior completa empleando el mismo número de línea para introducir la modificada.

Pero si queremos cambiar una línea larga ya introducida, el primer método no funciona y el segundo resulta laborioso. Lo mejor es editar (**EDIT**) la línea.

El cursor de programa

Veamos de qué se trata. Vamos a editar mi versión del programa sobre los círculos (Ejercicio 9.1 del capítulo 9). Usted puede probar con su propia versión, o bien utilizar la mía. Como prefiera.

Si observa el programa en la pantalla, verá que delante de uno de los números de línea hay un indicador o cursor [>], que denominamos indicador de línea o cursor de programa. A menos que usted lo haya movido, se encontrará delante de la última línea introducida. Lo primero que hay que hacer es llevar este indicador hasta la línea que deseamos editar:

(1) Si esa línea está próxima, podemos utilizar la

tecla ⇧ (**SHIFT** 7) o ⇩ (**SHIFT** 6) para moverlo en sentido ascendente o descendente, línea por línea.

(2) Para llevarlo al principio del programa, pulse **LIST** y **NEWLINE**. El indicador, que aparentemente ha desaparecido, se encuentra ahora frente a una imaginaria línea Ø y basta con tocar ⇩ para que baje una línea y reaparezca.

(3) Para situarlo en cualquier lugar teclee **LIST** y el mismo número de línea. En pantalla aparecerá una parte del programa, empezando por esa línea y con el indicador ante ella.

Practique estos tres métodos moviendo el cursor de programa arriba y abajo.

Edición de una línea

Quiero editar la línea 9Ø de mi programa:

9Ø **PRINT** "DIAM = ";D;" CM","CIRCUNF = ";C;" CM"

y borrar toda referencia al diámetro e imprimir "circunferencia" en **TAB** 3.

Primero pongo el cursor de programa frente a la línea 9Ø y presiono **EDIT** (**SHIFT** 1). La línea 9Ø aparece inmediatamente en la parte inferior de la pantalla, con el cursor [K] situado a continuación de la línea 9Ø. Ahora puedo mover el cursor [K] hacia delante o hacia atrás utilizando ⇦ (**SHIFT** 5) o ⇨ (**SHIFT** 8) sin que cambie el contenido de la línea. Pruébelo y vea como el cursor avanza o retrocede y cómo cambia a [L] cuando sobrepasa **PRINT**. Sitúelo justo después de la coma en el centro de la línea y entonces use **RUBOUT** para eliminar todo lo que esté detrás hasta las primeras comillas. Ahora tenemos:

9Ø **PRINT**[L] "CIRCUNF = ";C;" CM"

Teclee **TAB** 3 y entonces mueva el cursor hasta que

esté exactamente detrás de CIRCUNF, y teclee ERENCIA.

Si usted no acaba de estar satisfecho, pulse de nuevo **EDIT** y reaparecerá otra vez la línea original en la base de la pantalla.

Suponiendo que usted está conforme con la línea corregida:

```
9Ø PRINT TAB 3;"CIRCUNFERENCIA ";C;" CM"
```

presione **NEWLINE** e inmediatamente se situará en el lugar correspondiente del programa. La línea sustituida habrá desaparecido definitivamente.

Renumeración de líneas

Ahora queremos numerar de nuevo la línea 9Ø, asignándole el 1Ø5. Con el ZX81 no hay ninguna dificultad. Edite la línea pulsando **EDIT** y entonces presione dos veces **RUBOUT** para borrar el 9Ø. Teclee el nuevo número 1Ø5 y pulse **NEWLINE** para incorporar la línea al programa. La línea 9Ø permanece en su sitio. Para borrarla tendrá que teclear 9Ø **NEWLINE**.

Algunos puntos más

Recuerde que puede usar también las flechas para editar una línea que esté usted escribiendo por primera vez.

Si usted escribe un programa largo, sólo le cabrá una parte en la pantalla. El ZX81 le muestra precisamente la parte sobre la que usted está trabajando. No obstante, puede teclear **LIST** n para que aparezca la línea n y siguientes en la pantalla.

Cuando la memoria disponible de su ZX81 está próxima a agotarse, observará que **EDIT** no actúa, especialmente con líneas de programa largas. El remedio consiste en teclear **CLS** y **NEWLINE**. De este modo se borra la pantalla y **EDIT** vuelve a funcionar.

En este capítulo hemos aprendido ...

Comandos

EDIT para modificar una línea ya introducida en el programa.

LIST, LIST n para llevar a la pantalla diferentes partes de un programa largo.

Otros

El cursor de programa o indicador de línea y cómo desplazarlo verticalmente.

Cómo mover el cursor de línea.

Modificar la numeración de las líneas.

11

Funciones

En el teclado las funciones se encuentran bajo las teclas de letras, junto a algunas cosas que no son funciones. No se preocupe si echa de menos en este capítulo algunas funciones matemáticas. Consulte la lista de funciones al final del libro.

Una función de un número es una instrucción que realiza alguna operación con ese número y obtiene un resultado. El número sobre el que se opera (también puede ser una expresión o variable) se denomina 'argumento' de la función. Pruebe tecleando estos comandos:

PRINT SQR 81
LET A = 25 y luego **PRINT SQR** A
PRINT SQR 2
PRINT SQR (A*9)

Supongo que ha adivinado que SQR es nuestra vieja amiga la raíz cuadrada (un número que multiplicado por sí mismo da el número de partida). Observe que en el último ejemplo solicitamos la raíz cuadrada de una expresión; por esto la hemos puesto entre paréntesis. Una función opera siempre con el número o variable que le sigue inmediatamente, a menos que se le indique otra cosa mediante el empleo de paréntesis. En otras palabras, una función tiene mayor prioridad que cualquier otro operador matemático.

Podemos usar varias funciones conjuntamente. En este caso se ejecutan una por una de derecha a izquierda. Por ejemplo:

PRINT LN SQR 16

nos da el logaritmo neperiano (en base "e") de la raíz cuadrada de 16. El ZX81 trabaja exclusivamente con logaritmos neperianos (o naturales), así como con antilogaritmos neperianos (**EXP** o e^x).

Las funciones trigonométricas

Tome cualquier círculo, divida la circunferencia por el diámetro y obtendrá siempre un mismo número, ligeramente superior a 3, que llamamos PI y simbolizamos por la letra griega π. Si quiere mayor precisión teclee:

PRINT PI (π en el teclado)

Todas las funciones trigonométricas están relacionadas con ángulos, los cuales hay que expresar en radianes para que el ZX81 los entienda. Podemos convertir fácilmente grados en radianes si recordamos que:

PI radianes = 180º

Si le interesa este tema practique con el siguiente programa que corresponde a una pequeña tabla trigonométrica:

```
10  LET XD = 30
20  REM XD ES EL ANGULO EN GRADOS
30  LET XR = XD*PI/180
40  REM XR AHORA ESTA EN RADIANES
50  PRINT XD;" GRADOS",XR;" RADIANES"
60  PRINT ,, "SENO = ";SIN XR
70  PRINT ,, "COS = ";COS XR
80  PRINT ,, "TAN = ";TAN XR
```

(vea las ,, para espaciado de línea en 60, 70 y 80).

Ejecute el programa y haga pruebas introduciendo diferentes ángulos en la línea 1∅. Compruebe los resultados con la ayuda de unas tablas trigonométricas. Anote alguno de los conjuntos, como éste:

```
6∅      SEN = ∅.866∅254      COS = ∅.5
        TAN = 1.732∅5∅8
```

Ahora teclee el comando:

PRINT ASN ∅.866∅254*18∅/**PI**

y volverá a tener el ángulo original de 60º. **ASN** X (en el teclado, ARCSIN) da el ángulo en radianes cuyo seno es X. ARCCOS y ARCTAN hacen lo mismo con el coseno y la tangente.

Aquí hay un hacha

Más funciones: **INT**, **ABS** y **SGN**. Estudiémoslas practicando:

```
1∅  LET N = 3
1∅∅ PRINT "NUMERO";TAB 8;"ENT","ABS";TAB 24;"SGN"
11∅ PRINT
12∅ PRINT N;TAB 8;INT N,ABS N;TAB 24;SGN N
```

Asigne toda clase de números a N en la línea 1∅: enteros, decimales, negativos. Si le da pereza, aquí tiene unos ejemplos:

NUMERO	ENT	ABS	SGN
3	3	3	1
3.14	3	3.14	1
∅.14	∅	3.14	1
∅	∅	∅	∅
-3	-3	3	-1
-3.14	-4	3.14	-1

Resulta obvio que **INT** "corta" y desprecia la parte decimal de un número, respetando la parte entera, que es inferior al número original.

3.14 da 3 (el entero inmediato inferior a 3.14)
-3.14 da -4 (el entero inmediato inferior a -3.14)

ABS es otra "cuchilla" que elimina cualquier signo negativo y lo reemplaza por un signo positivo. Es decir, da el valor absoluto de N.

Empuñemos el hacha de nuevo con **SGN**. Ahora ha desaparecido el número completo y lo único que queda es el signo, + o -, asociado a un 1. Cero no tiene signo, luego **SGN** Ø = Ø.

Ejercicio 11.1 Parte decimal

No existe una función que genere la parte decimal de un número. Eso tendrá que hacerlo usted. Escriba un programa que imprima un número, su parte entera y su parte decimal en sendas columnas.

Redondeo de números

Los ordenadores dan a menudo los resultados con tantos decimales que resultan engorrosos. Por lo tanto es frecuente la necesidad de redondear. **INT** no lo hace por sí misma. Para ver por qué teclee:

PRINT INT 7.Ø1 y **PRINT INT** 7.99

Ambos dan 7, lo cual no es correcto en el caso de 7.99, que está muy próximo a 8. Lo adecuado es sumar 0.5 al número antes de aplicar **INT**, así:

N	N + Ø.5	INT (N + Ø.5)
7.Ø1	7.51	7
7.49	7.99	7
7.5	8.0	8
7.99	8.49	8

Ejercicio 11.2 Redondeo de decimales

Hemos aprendido a redondear números hasta el entero más próximo. Ahora escriba un programa que redondee un número hasta la primera cifra decimal. Le sugiero que multiplique por 1Ø, redondee el resultado hasta el próximo entero y entonces divida por 1Ø. Pruébelo:

Espero que le haya salido bien. Por el mismo procedimiento puede redondearse un número hasta cualquier cifra decimal.

Aún queda algo importante. El ZX81 necesita enteros para ejecutar ciertas instrucciones. Ya hemos visto **TAB** n y **LIST** n. También hay otras:

PLOT	**UNPLOT**	**RUN**	**DIM**	**GOTO**
GOSUB	**PAUSE**	**PRINT**	**AT**	**PRINT** (**TO**)

Puede utilizar variables o expresiones con estas sentencias. El ZX81 calculará los valores. El problema es que las expresiones y variables con frecuencia conducen a números decimales. No se preocupe. El ZX81 los redondeará hasta el entero más cercano, igual que hicimos al principio de esta sección.

En este capítulo hemos aprendido ...

Funciones: matemáticas, trigonométricas, **INT**, **ABS** y **SGN**.

Redondeo de números.

Redondeo automático por el ZX81, cuando alguna instrucción necesita enteros (**LIST** n por ejemplo).

12

La rueda mágica

¿Se acuerda del Aprendiz de Brujo del capítulo 3?. Aquí tenemos un modelo matemático de escoba que llena de agua un tanque de 15Ø litros a razón de cuatro litros por viaje. Necesitamos espacio en pantalla para ver las líneas correspondientes a un buen número de viajes. Con este objeto introducimos una nueva instrucción, **SCROLL**, que mueve ascendentemente el contenido de la pantalla, línea por línea, dejando espacio en la parte inferior para que podamos ir tecleando. Introduzca este programa:

```
1Ø  LET  W = Ø
2Ø  LET  W = W+4
3Ø  SCROLL
4Ø  PRINT  W; " LITROS"
5Ø  GOTO  2Ø
```

- La línea 1Ø vacía el tanque para empezar.
- La 2Ø vierte 4 litros de agua en el tanque.
- Las líneas 3Ø y 4Ø imprimen la cantidad total de agua vertida hasta el momento.
- La línea 5Ø contiene una instrucción muy importante. **GOTO 2Ø** significa: "Ve a la línea 2Ø y continúa ejecutando el programa desde allí". En otras palabras: "Ve otra vez a la fuente a buscar más agua".

¿Puede usted predecir el resultado de este programa?. Ejecútelo y observe... ¡agua, agua por todas partes!. ¿Pero cómo podemos parar las escobas?. **BREAK** (abajo a la derecha; no es preciso pulsar **SHIFT**) será nuestro recurso de emergencia. Sirve para detener el ZX81 mientras está trabajando, así que utilícelo ahora. Si desea reanudar la ejecución tras haber pulsado **BREAK**, toque **CONT**; en cualquier caso el contenido de la pantalla se habrá borrado.

Habíamos creado un bucle entre las líneas 2Ø y 5Ø. ¡**GOTO** es un recurso de un valor inapreciable para realizar grandes cantidades de trabajo!. En el capítulo 3 ya vimos que precisábamos un salto condicional dentro del bucle para comprobar si el tanque está lleno. Hagámoslo ahora cambiando la línea 5Ø y añadiendo dos líneas más:

5Ø **IF** W < 15Ø **THEN GOTO** 2Ø
6Ø **SCROLL**
7Ø **PRINT** "TANQUE LLENO. ¿QUE HAGO AHORA?"

Todo ha ido bien, ¿verdad? (aparte de haber derramado dos litros). La línea 5Ø contiene una pieza fundamental en programación. Equivale a decir: 'Comprueba el valor de W. Si es menor que 15Ø, vuelve a la línea 2Ø. Si es igual o mayor que 15Ø, pasa a la línea siguiente'. ¡El BASIC no despilfarra las palabras!.

Operadores relacionales

La línea 5Ø tiene la forma general:

IF (si se cumple que ...) **THEN** (haz esto otro).
Ejemplo: W < 15Ø Ejemplo: **GOTO** 2Ø

A continuación de la palabra clave **IF** siempre hay una instrucción que incluye uno de los operadores relacionales que usamos para efectuar comparaciones:

= igual que
< menor que

\> mayor que
<= menor o igual que
\>= mayor o igual que
<> distinto que

A ambos lados de los operadores ponemos los términos que se comparan. Estos pueden ser números, variables o expresiones:

```
IF A = 1ØØ THEN ...
IF B < Ø THEN ...
IF C > A + 21 THEN ...
IF D <> A + B THEN ...
```

Sí (IF) algo es cierto, entonces (**THEN**) ¿qué?

En el programa anterior vemos:

... **THEN GOTO** 2Ø

que es una forma muy corriente de instrucción condicional. Sin embargo **THEN** mantiene el cursor [K] (¿lo había notado?) y puede ser seguida por cualquier otra palabra clave, aunque algunas de ellas no tendrían sentido. Son típicas éstas:

GOTO **PRINT** **LET**
GOSUB **RETURN** (esta la veremos más adelante)

He aquí algunos ejemplos de línea condicional:

```
1ØØ  IF X > 21 THEN PRINT "MAS DE 21. PIERDE"
2ØØ  IF T >= Z THEN GOTO 1ØØØ
3ØØ  IF P < Ø THEN LET P = Ø
```

Ve a (**GOTO**) ¿dónde?

Tanto si **GOTO** es imperativo o condicional, tiene que ser seguido de un número de línea. Así puede dirigir al ZX81 a cualquier línea del programa, anterior o posterior. Podemos emplear un número de línea como tal

o bien una expresión (con todas las variables previamente definidas, por supuesto). Si el resultado es un número decimal, el ZX81 lo redondeará para convertirlo en entero, y si el número de línea obtenido no existe, el ZX81 pasará al siguiente.

¿Y el **STOP**?

En medio de este galimatías puede sernos muy provechoso conocer cómo parar. Pruebe con este programa:

```
1ØØ LET S = 78
2ØØ IF S > = 1ØØ THEN GOTO 4ØØ
3ØØ PRINT "PIERDE. PUNTUACION =  ";S
4ØØ PRINT "GANA. PUNTUACION = 1ØØ + "
```

Si lo ejecuta verá que necesita algo para detener al ZX81 que, de lo contrario, ejecutaría ambas instrucciones 3ØØ y 4ØØ.

```
35Ø STOP
```

Añada esta instrucción y todo irá bien. Pruebe el programa con distintos valores de S y asegúrese de que funciona.

Ahora resuelva estos dos problemas. Los dos necesitan bucles **IF... THEN.**

Ejercicio 12.1 Constructora

Una constructora le ofrece un interés compuesto de un 8% con periodo anual. Si usted deposita 5ØØ £ y permite que se les vayan sumando los intereses, al cabo de un año tendrá:

$$5ØØ \times \frac{1Ø8}{1ØØ} = 54Ø \text{ £}$$

A los dos años:

$$54Ø \times \frac{1Ø8}{1ØØ}$$ y así sucesivamente.

Escriba un programa que muestre cómo van creciendo sus ahorros a lo largo de 7 años, finalizando en 1989. Luego cambie una línea para redondear cada resultado hasta el penique más próximo (ver capítulo 11).

Ejercicio 12.2 ¿Cuáles son los años bisiestos?

Para saber si un año es bisiesto hay que comprobar si sus dos últimas cifras son divisibles por 4. Escriba un programa que imprima los años desde 1982 a 1999 y diga cuáles son los bisiestos. La tabla siguiente le ayudará:

Año Y	Año dividido por 4: (Y/4)	INT (Y/4)	¿Es INT (Y/4) igual a Y/4?
1982	2Ø.5	2Ø	NO
1984	21	21	SI

En este capítulo hemos aprendido ...

Comandos

BREAK para detener al ZX81 mientras trabaja.

CONT para reiniciar después de **BREAK**.

Instrucciones

SCROLL para mover hacia arriba el contenido de la pantalla, línea por línea, para ir dejando espacio para escribir en la base.

GOTO n dirige al ZX81 a la línea n del programa.

IF condición **THEN** instrucción, ejecutan la instrucción dada (**GOTO**, etc.) si se cumple la condición establecida (X = 1Ø, etc).

STOP para detener la ejecución del programa e impedir que ejecute líneas indebidamente.

Otros

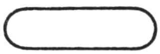
Operadores relacionales (= , > , < , >=, <=, <>) para efectuar comparaciones.

13

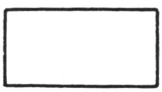
Diagramas de flujo

Ahora que ya conocemos los bucles y las instrucciones condicionales tenemos la posibilidad de escribir programas bastante complicados. Es hora de abordar los diagramas de flujo como ayuda a la programación.

Suponga que quiere programar determinada operación (usemos el caso del Aprendiz de Brujo del capítulo 3 como ejemplo). La idea central de un diagrama de flujo consiste en dividir la operación en fases independientes, escribir cada fase en una figura y unir todas las figuras con flechas para mostrar el orden de ejecución de las fases. Las figuras que utilizamos son las siguientes:



Principio o final



Símbolo de "proceso". Una fase de la operación que no incluya decisiones.

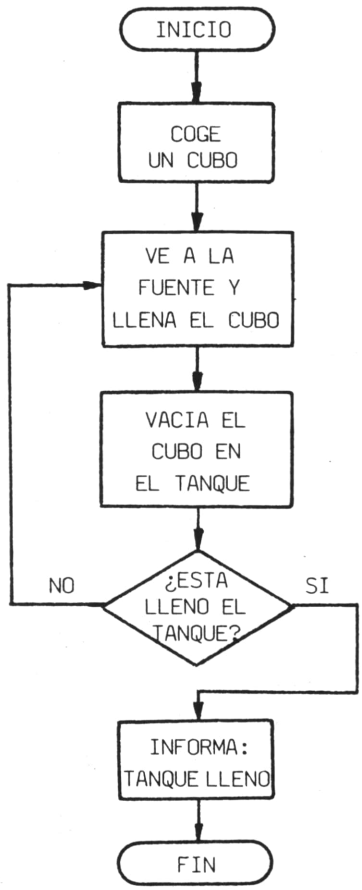


Símbolo de "decisión". Se plantea una pregunta y se establece una bifurcación. El camino que se tome dependerá de la respuesta a la pregunta.

Ahora dibujemos un diagrama de flujo para llenar el tanque con agua de la fuente. Compárelo con el programa original del capítulo 3 y con el modelo matemático del último capítulo. Observe que los rombos representan las instrucciones IF... THEN.

Hay personas capaces de escribir directamente los programas sin necesidad de diagrama de flujo. No obstante a la mayoría de nosotros nos será útil. Más adelante veremos otros ejemplos de diagramas de flujo.

Escoba llenando el tanque con agua de la fuente



14

Introducción de datos

Volvamos al ejercicio 12.1 del capítulo 12 (encontrará el programa en el apéndice 4). No es un mal programa. Le da el interés tras una inversión de 5ØØ £ al 8 % durante 7 años. Pero resulta engorroso cuando se quiere modificar el capital: hay que teclear de nuevo la línea 2Ø. Bueno, no importa, podemos mejorarlo. Simplemente cambie la línea 2Ø para que diga:

```
2Ø  INPUT C
```

Ejecute el programa. ¿Pero qué es esto?. ¡La pantalla en blanco y un cursor [L] en la base!. El ZX81 está diciéndonos: "Me he detenido y estoy esperando que me introduzcas un valor para la variable C". Teclee 5ØØ y presione **NEWLINE**. Obtendrá exactamente el mismo resultado que antes con:

```
2Ø LET C = 5ØØ
```

pero ahora usted puede asignar a C un valor diferente cada vez que ejecute el programa. Pruébelo. Verá que **INPUT** es estupenda.

Para facilitar las cosas a otras personas que utilicen sus programas, puede imprimir una línea recordándoles qué tipo de dato deben introducir:

```
10 PRINT "¿CUAL ES SU CAPITAL?"
```

¿Fácil, verdad?. No se vaya; aún hay más cosas.

Bucles con INPUT

Tras haber ejecutado el programa, suponga que desea volver al principio y repetir con capitales distintos. ¿Qué instrucción usaría?. Pues sí, lo ha adivinado:

```
150 GOTO 10
```

Tecléela y ejecute con diferentes valores de C. ¡Vaya, no vale!. Se ha detenido con un código 5/100. Si mira el manual verá que significa "pantalla llena". Habrá que hacer algo al respecto. Por supuesto podemos recurrir a **SCROLL**, pero ¿imagina usted los resultados desfilando pantalla arriba con una pausa de vez en cuando?. Está claro que en este caso es más elegante borrar la pantalla antes de cada impresión. La instrucción para hacerlo es **CLS** (clear screen). Vea el listado completo:

```
10  PRINT,,"¿CUAL ES SU CAPITAL?"
20  INPUT C
30  CLS
50  LET Y = 1982
100 PRINT Y;"CAPITAL + INTERES =  £";C
110 PRINT
120 LET Y = Y+1
130 LET C = C*1.08
140 IF Y < 1990 THEN GOTO 100
150 GOTO 10
```

Ahora tenemos dos bucles, uno dentro de otro (anidados). ¿Y por qué no podemos poner **CLS** al principio del bucle exterior?. Pruebe y lo verá.

Salir de un bucle con INPUT

¿Pero cómo se para esto?. ¡Parece que va a estar pidiéndonos datos eternamente!. El modo de detenerlo es pulsar **STOP** en una pausa input; acto seguido con

NEWLINE se suspenderá la ejecución y obtendremos un mensaje D/2Ø. Si desea reanudar toque **CONT**; el ZX81 estará todavía en la línea 2Ø esperando nuevos datos, aunque **CONT** habrá borrado la pantalla.

Bucles permanentes

El anterior es un bucle permanente que resulta posible tan sólo gracias a que la instrucción **INPUT** va haciendo pausas en la línea 2Ø. Si la sustituimos por:

```
2Ø LET C = 5ØØ
```

veremos que el pobre ZX81 gira y gira siguiendo el bucle externo indefinidamente (o hasta que presionemos **BREAK**). ¡Un mal programa!.

Ahora, unos ejemplos para que practique los bucles con **INPUT**.

Ejercicio 14.1 Porcentajes

Escriba un programa para expresar sus calificaciones en los últimos exámenes en porcentajes. Vaya introduciendo sus notas y la nota máxima posible y utilice la fórmula:

$$\frac{\text{Nota}}{\text{Nota máxima}} \times 1ØØ = \%$$

Ejercicio 14.2 Consumo de gasolina

Haga un programa para introducir los Km recorridos y los litros de gasolina consumidos y que luego calcule los Km por litro.

En este capítulo hemos aprendido...

Instrucciones

INPUT para detener el programa e introducir valores de variable.

CLS para limpiar la pantalla.

STOP como input para salir de un bucle.

CONT para reanudar después de **STOP**.

Otros

Bucles con input para detener repetidamente la ejecución e ir introduciendo datos.

15

Guardar programas y datos

La impresora ZX

Imagine que ha escrito un buen programa y que desea conservarlo. Cuando desconecte el ZX81 el programa se borrará y usted tendrá que desarrollarlo de nuevo. Por supuesto puede anotarlo en un papel, pero es un trabajo engorroso y es fácil cometer errores. Todo será más sencillo si posee una impresora Sinclair ZX y ha tenido la buena idea de conectarla al ZX81 antes de empezar (Sinclair recomienda no conectar la impresora sin desenchufar primero). En este caso podrá listar en papel su programa tecleando el comando **LLIST**. Si sólo necesita guardar una parte del programa puede pulsar **LLIST** n y obtendrá un listado desde la línea n en adelante. Además puede suspender la impresión en el punto que quiera tocando **BREAK**.

Otro modo de emplear la impresora consiste en hacer una copia sobre papel de la salida por pantalla. Use la palabra clave **COPY**, ya sea como comando o como instrucción, para imprimir el contenido de la pantalla.

Back-up

Tanto si ha copiado el programa a mano como si lo ha impreso en papel con la impresora ZX, tendrá que teclear bastante cuando quiera utilizarlo de nuevo. Puede ahorrarse todo ese trabajo haciendo lo que se conoce como <u>back-up storage</u>, es decir, guardar el programa en algún soporte susceptible de ser leído por el



El ZX81 y la impresora ZX Printer diseñada especialmente para ser utilizada con los ordenadores personales Slinclair.

ZX81 sin necesidad de introducción por teclado. Esto podemos realizarlo auxiliándonos con casi cualquier tipo de grabadora de cinta magnética.

Guardemos un programa

Mi cassette tiene conectores de 3.5 mm para micrófono (MIC), auricular (EAR) y ajuste automático del nivel de grabación. Dispone de un buen contador (muy útil) y un indicador de nivel de grabación. Si a su aparato le falta alguna de estas características, seguramente servirá, pero será más incómodo.

Necesitará una cinta dedicada exclusivamente a programas del ZX81 y deberá anotar cuidadosamente su contenido. A continuación le indico una serie de operaciones que a mí me han dado buen resultado (si tiene problemas consulte el capítulo 16 del manual de instrucciones del ZX81):

(1) Conecte entre sí los terminales señalizados con MIC en el ZX81 y en el cassette.

(2) Rebobine la cinta hasta el principio, ponga a cero el contador, bobine la cinta hasta que pase todo lo que hay ya grabado y anote la lectura del contador.

(3) Si quiere puede grabar de viva voz a través del micrófono el nombre del programa. Esto es conveniente cuando el aparato no tiene contador.

(4) Pulse el comando **SAVE** "NOMBRE". Elija e introduzca el nombre. Tome nota del mismo.

(5) Presione las teclas RECORD y PLAY del cassette. Entonces toque **NEWLINE**. Tras cinco segundos en blanco verá un conjunto de bandas blancas y negras (su programa). Compruebe mediante el indicador de nivel que la grabación se está efectuando.

(6) Al final la pantalla quedará en blanco con un código Ø/Ø. Desconecte el cassette. El programa permanece intacto en el ZX81.

Es prudente dejar algún espacio de separación entre los diversos programas ya grabados; cinco segundos bastarán. Un programa de 1K tarda de 15 a 20 segundos en grabarse.

Cargue su programa

Ha pasado un día y usted quiere introducir de nuevo su programa de ayer en el ZX81. Yo lo hago así:

(1) Bobino la cinta hasta el punto donde empezaba la grabación.

(2) Conecto los terminales EAR del cassette y del ZX81.

(3) Tecleo **LOAD** "NOMBRE" o simplemente **LOAD** " ".

(4) Fijo el mando de volumen del cassette a 3/4 del máximo aproximadamente, y los mandos de tono,

si los hay, en el máximo de agudos y mínimo de graves.

(5) Presiono **NEWLINE**. Ahora se ven una serie de figuras en la pantalla y después un rápido baile de franjas horizontales, algo así como una persiana que se haya vuelto loca. Es el programa.

(6) Después de la carga, la pantalla aparece limpia y con un código Ø/Ø. Entonces desenchufo el cassette.

(7) Ahora puedo presionar **NEWLINE** para listar el programa o bien **RUN** para ejecutarlo.

Es preferible cargar programas con nombre

Uno se vuelve perezoso y deja de teclear el nombre del programa entre las comillas que siguen a **LOAD**. Sin embargo esto hace la operación de carga menos fiable. Si usted ha dado nombre al programa, el ZX81 ignorará todos los demás, incluso la cola de algún programa anterior. De hecho, si es necesario, el ZX81 buscará en la cinta y cargará el programa cuyo nombre se haya dado. Hay que dar el nombre exacto; una letra o espacio equivocado y no se cargará nada.

Guardar datos

Muchos ordenadores utilizan instrucciones DATA y READ que permiten escribir instrucciones con muchos datos. El ZX81 no dispone de esta facilidad y además introducir datos mediante instrucciones **LET** o **INPUT** es verdaderamente tedioso.

¡Pero no todo está perdido!. Es importante darse cuenta de que una vez que se ha ejecutado un programa e introducido un dato a través de **INPUT**, sólo hay tres operaciones que borrarán los datos:

(1) Desenchufar el ZX81.

(2) Presionar **RUN** otra vez.

(3) Pulsar **CLEAR** (una tecla que borra todas las variables).

Para hacer trabajar el programa sin tocar **RUN** tenemos que emplear **GOTO** como comando. Teclee este breve programa:

```
1Ø  INPUT A
2Ø  INPUT B
3Ø  INPUT C
1ØØ PRINT A;B;C;" GO"
```

Ahora teclee el comando **GOTO** 1ØØ. Obtendrá el código 2/1ØØ que significa: "variable desconocida". Ejecute (**RUN**) el programa y asigne los valores 1, 2, 3 a las variables A, B y C. Esta vez tendrá la salida esperada:

123 GO

Si a continuación introduce el comando **GOTO** 1ØØ conseguirá exactamente el mismo resultado (¡los datos permanecen en su sitio!). Sin otras manipulaciones, guarde (**SAVE**) el programa del modo usual y, con él, habrá guardado también los datos. Para comprobarlo primero desenchufe un momento el ZX81 y así descartará la posibilidad de hacer trampas. Entonces efectúe la carga del modo usual, pulse el comando **GOTO** 1ØØ y el resultado:

123 GO

confirmará que los datos siguen en su lugar. Si en cualquier momento toca **RUN**, los datos desaparecen y tendrá que introducirlos de nuevo. Un último punto: si va escaso de espacio en memoria, puede economizar algunos bytes así:

(1) Escriba la parte del programa imprescindible para cargar los datos.

(2) Ejecútela e introduzca los datos.

(3) Borre las líneas de programa escritas hasta

ahora y escriba la parte del programa que utiliza los datos.

(4) Guarde el programa y los datos y use **GOTO** n para ejecutar el programa, ¡nunca **RUN**!.

En este capítulo hemos aprendido ...

<u>Comandos</u>

LLIST o **LLIST** n para listar un programa sobre papel utilizando la impresora ZX.

COPY para copiar sobre papel el contenido de la pantalla mediante la impresora ZX. También puede emplearse como instrucción en el seno de un programa.

SAVE para transferir programas desde el ZX81 a una cinta magnética.

LOAD para cargar programas en el ZX81 desde la cinta magnética.

GOTO n para ejecutar un programa a partir de la línea n sin borrar ningún dato.

<u>Otros</u>

Guardar datos en cinta y cargarlos de nuevo.

16

Gira y gira... diez veces

Simplemente ojeando de nuevo el capítulo 12, usted debe ser capaz de hacer que un bucle se recorra exactamente diez veces. ¿Verdad que sí?:

```
10  LET J = 0
20  LET J = J+1
30  PRINT J;" VECES EL BUCLE"
40  IF J<10 THEN GOTO 20
50  PRINT
60  PRINT "PARADO PARA DESCANSAR"
```

El BASIC tiene una instrucción especialmente concebida para hacer lo mismo. Cambie el programa anterior así:

```
Borre la línea 10
20  FOR J = 1 TO 10
40  NEXT J
```

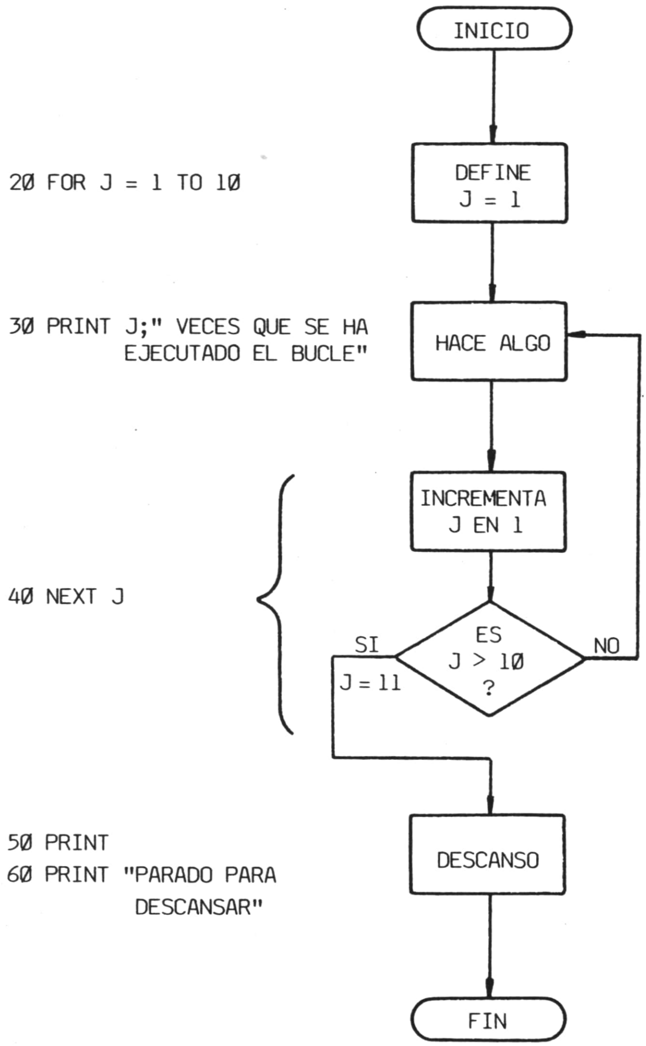
Ejecútelo. El resultado será idéntico. **FOR/NEXT** es un medio excelente de ahorrar tiempo. Para comprender como trabaja estudie detenidamente el programa y el diagrama de flujo de la página siguiente.

He aquí algunas notas sobre los bucles **FOR/NEXT**:

(1) N es la <u>variable de control del bucle</u>. Puede consistir en una letra, de la A a la Z, pero

evite las letras que use en otro lugar como variables ordinarias.

(2) Los números a ambos lados de **TO** son los límites inferior y superior para la variable de control del bucle. Pueden ser números, variables o expresiones, pero el ZX81 no redondea los valores en este caso.



(3) La variable de control se incrementa en 1 cada vez que es recorrido el bucle. Fíjese que termina en 1 unidad por encima del límite superior.

(4) El bucle puede incluir cualquier número de líneas de programa con cualquiera de las instrucciones usuales. Puede hacer lo que desee, incluso utilizar el valor de J, mientras no lo cambie. Recuerde que N se incrementa en 1 cada vez que se completa el bucle.

(5) Es posible salir del bucle con una instrucción **IF/THEN GOTO**, pero no trate de entrar en un bucle **FOR/NEXT** si no lo hace por el principio; el ZX81 no sabría cuál es la variable de control.

6) Un **FOR** sin el correspondiente **NEXT** es incorrecto y será ignorado. Un **NEXT** sin **FOR** previo detendrá la ejecución del programa y dará un mensaje de error 1/n o 2/n.

Ahora pruebe la técnica **FOR/NEXT** con este problema:

<u>Ejercicio 16.1</u> Tabla de raíces cuadradas

Escriba un programa para imprimir los números enteros, desde el Ø al 16, con su raíz cuadrada al lado y bajo un encabezamiento apropiado.

Paso a paso

¡Agárrese fuerte antes de leer esto!. No tenemos por qué andar sumando 1 cada vez que se completa un bucle. Si añadimos la palabra mágica **STEP**, podemos incrementar en cualquier valor, o incluso decrementar empleando intervalos negativos. Vea unos ejemplos:

```
FOR N = 1 TO 12 STEP 3
FOR J = 8 TO Ø STEP -1
FOR K = P TO Q STEP 3*R
FOR L = Ø TO 5 STEP Ø.5
```

Pruebe cambiando la línea adecuada en su último programa sobre raíces cuadradas:

FOR N = Ø **TO** 16 **STEP** 2

y

FOR N = 16 **TO** Ø **STEP** -2

Ahora practique **FOR/NEXT/STEP** en este ejercicio:

Ejercicio 16.2 Múltiplos

Escriba un programa que obtenga los múltiplos de 4 entre Ø y 1ØØ en cuatro columnas. Una sugerencia: utilice el valor de la variable de control del bucle no sólo como múltiplo de 4, sino también como indicador de la columna donde debe imprimirse.

En este capítulo hemos aprendido ...

Instrucciones

FOR/TO/NEXT para ejecutar un bucle un número determinado de veces.

STEP para especificar los incrementos o decrementos de la variable de control del bucle cuando convenga que sean diferentes de +1.

17

Bucles dentro de otros bucles

En el último capítulo cada programa tenía un bucle **FOR/NEXT**. Sin embargo, el número de bucles posible en un programa no tiene límite. Véalo:

```
10  FOR J = 0  TO  4
20  PRINT TAB J; "PRIMER BUCLE"
30  NEXT J
40  PRINT
100 FOR J = 10 TO 12
110 PRINT TAB J; "SEGUNDO BUCLE"
120 NEXT J
```

Observe que hemos empleado J como variable de control de ambos bucles. Esto es válido en el caso de bucles separados y es útil para economizar memoria.

```
10  FOR J = 1 TO 3
20  PRINT "BUCLE EXTERNO"
30  FOR K = 1 TO 5
40  PRINT TAB 5; "BUCLE INTERNO"
50  NEXT K
60  NEXT J
```

Esta vez tenemos un "bucle K" dentro de un "bucle J". A esto lo llamamos "bucles en series internas" o "bucles anidados". Puede utilizar hasta 26 bucles anidados, pero debe respetar dos importantes reglas:

(1) Tiene que usar letras distintas como variables de control de los bucles.

(2) Los bucles internos deben estar contenidos en su totalidad dentro de los externos, no solapados parcialmente.

¿Se acuerda de las tablas de multiplicar en forma de matriz?. Aquí tiene un ejemplo:

0	1	2
1	2	3
2	3	4

Si traza una línea descendente desde el número 1 de la primera fila y otra línea horizontal desde el 2 de la columna izquierda, ambas líneas se cortarán en el 3, mostrando que 1+2 = 3. Estos cuadros pueden construirse fácilmente con el ZX81:

```
10 FOR J = 0 TO 5
20 FOR K = 0 TO 5
30 PRINT TAB 4*K;J+K;
40 NEXT K
50 PRINT,""
60 NEXT J
```

Y ahora un ejercicio:

Ejercicio 17.1 Tabla de multiplicar

Aquí tiene una tabla de multiplicar similar a la anterior:

1	2	3
2	4	6
3	6	9

Modifique el último programa para escribir una tabla que cubra los números de 1 a 7.

Gráficos sencillos

Gráficos y bucles a menudo van juntos, así que vamos a echar una ojeada a los elementos gráficos del teclado. Recuerde que tiene que pulsar **GRAPHICS** al empezar y al terminar. Cada bloque o elemento gráfico es un pequeño cuadrado del tamaño de una letra, dividido en cuatro cuadraditos, que pueden ser negros, blancos o, hasta cierto punto, grises. Están bien ilustrados en la página 78 del manual del ZX81. Además disponemos de impresión inversa de letras, algunos símbolos y espacio (cuadrado negro) que tienen gran utilidad. Podemos hacer que el ZX81 imprima cualquiera de los bloques gráficos como si fueran letras. Aquí tiene un cuadrado gris de 8 por 8 a título de ejemplo:

```
1ØØ PRINT "▨";
```

Teclee y ejecute. Sale un cuadrado, pero ¡faltan 63!. Añada estas líneas:

```
9Ø  FOR K = 1 TO 8
11Ø NEXT K
```

¡Perfecto!. Ya tenemos una fila de 8 bloques. Escriba dos instrucciones más para obtener 8 de estas líneas:

```
8Ø  FOR J = 1 TO 8
13Ø NEXT J
```

Ahora tenemos los 64 bloques, pero no exactamente en forma de cuadrado. ¿Qué es lo que está mal?. ¡Ah, sí!, es el punto y coma tras cada bloque el que hace que se impriman en fila. Tenemos que saltar de línea cada 8 bloques, es decir, tras cada bucle J. Una instrucción más y quedará bien:

```
12Ø PRINT
```

ahora el cuadrado es perfecto, y todo gracias a los bucles anidados.

Y ahora le toca a usted:

Ejercicio 17.2 Rectángulo

Escriba un programa que dibuje un rectángulo negro de 19 bloques de base por 5 de altura. Luego modifíquelo de tal manera que imprima ESTO ES UN RECTANGULO en modo inverso y en el centro del rectángulo.

Hay mucho más sobre gráficos. Volveremos a ellos en el capítulo 23.

En este capítulo hemos aprendido ...

Bucles múltiples y anidados.

Gráficos sencillos utilizando los bloques gráficos.

18

¡Qué máquina más simpática!

¿Se acuerda de **LET** y de **INPUT**?. Eran dos instrumentos para asignar valores a una variable numérica. ¡Sería estupendo poder hacer lo mismo con palabras!. Pues estamos de suerte, podemos hacer eso y más. Vea una muestra:

```
10  PRINT "TECLEE SU NOMBRE Y NEWLINE"
20  INPUT A$
30  PRINT "GRACIAS";A$
40  FOR J = 1 TO 200
50  NEXT J
60  PRINT,, "ES UN NOMBRE MUY BONITO"
```

¡A$ es la gran noticia!. Se trata de una variable de cadena. La ejecución del programa se detiene en la línea 20 y el cursor [L] en la base de la pantalla nos informa que el ZX81 está esperando que introduzcamos una cadena de caracteres. Por tanto, tecleamos tantos caracteres como deseemos (o ninguno, si se trata de una cadena vacía), presionamos **NEWLINE** y nuestra cadena queda asignada al nombre A$. Podemos utilizar A$ siempre que queramos, tal como hemos hecho en la línea 30. Las líneas 40 y 50 forman un bucle **FOR/NEXT** vacío; es un pequeño truco para establecer una pausa entre una salida y la siguiente. Nos es posible utilizar hasta 26 variables de cadena asociadas a nombres de una sola letra, de la A a la Z, seguida del signo $.

Si añadimos dos líneas más:

```
7Ø  GOTO 1Ø
25  CLS
```

tendremos un bucle para introducción de cadenas y podremos ir tecleando tantos nombres bonitos como queramos. Salir de uno de estos bucles puede resultar difícil, puesto que el ZX81 aceptará como cadena cualquier cosa que se teclee. Pruebe introduciendo **STOP**, por ejemplo. La solución consiste en quitar las comillas de la base de la pantalla pulsando **EDIT** (lo más fácil) o bien borrándolas del modo ordinario. Si ahora toca **STOP** y **NEWLINE** volverá a estar en posición de introducir comandos.

Aquí tiene un programa que utiliza **LET** para definir dos variables de cadena.

```
1Ø  LET A$ = "REGENT"
2Ø  LET B$ = "STREET"
3Ø  LET C$ = A$+B$
4Ø  LET N = 1Ø
5Ø  PRINT "¿QUIEN VIVE EN ";N;C$;"?"
```

En la línea 3Ø reunimos dos variables de cadena (concatenamos, solemos decir) y asignamos el conjunto a otro nombre de variable. En la línea 5Ø imprimimos toda una serie de datos: cadenas de caracteres, variables de cadena y números. Podemos imprimir cualquiera de ellos donde queramos dentro de la línea, usando ; , o bien **TAB**.

¿Y qué podemos hacer con variables de cadena?

Como acabamos de ver, podemos imprimirlas tantas veces como queramos y unirlas como si fueran cuentas de un collar utilizando + (- no actúa). También puede modificarlas como si se tratase de variables numéricas. Por ejemplo:

```
25 LET A$ = "DOWNING"
```

¡Hemos cambiado el contenido de A$!

Otra cosa que se puede hacer es comparar variables de cadena entre sí o con cadenas de caracteres, mediante nuestra vieja amiga **IF**. Más líneas:

```
60  INPUT P$
70  PRINT,,P$
80  IF P$ = "EL PRIMER MINISTRO" THEN GOTO 100
90  GOTO 50
100 PRINT,, "BIEN"
```

En la línea 80 empleamos el signo = para comparar la respuesta, introducida como P$, con la cadena "EL PRIMER MINISTRO". Recuerde que en BASIC = significa "exactamente igual"; ¡letras, espacios y signos de puntuación deben ser idénticos!. Ejecute el programa y modifique P$ para comprobarlo. Luego borre la línea 90 del programa y añada:

```
80  IF P$ <> "EL PRIMER MINISTRO" THEN GOTO 50
```

Ocasionalmente recurrimos a > y < para comparar variables de cadena. El siguiente programa muestra lo que ocurre:

```
10  PRINT "TECLEA UNA PALABRA"
20  INPUT A$
30  PRINT "AHORA OTRA"
40  INPUT B$
50  PRINT,,A$;"ESTA";
60  IF A$ > B$ THEN GOTO 100
70  PRINT "ANTES";
80  GOTO 110
100 PRINT "DESPUES";
110 PRINT B$, "EN EL DICCIONARIO"
```

Ejecute el programa introduciendo ARCO y ZOO primero, y luego ABRACADABRA y AARDVARK. Ahora ya sabe lo que significa > cuando se aplica a cadenas.

Con este capítulo nuestras posibilidades de programación han dado un gran paso adelante. Aquí tiene un programa sencillo para que practique:

Ejercicio 18.1 Rellenar un formulario

Escriba un programa que pregunte a alguien por su nombre, edad y lugar donde vive. Imprima la información y dé las gracias cortésmente.

De nuevo la impresora

Ahora que sabemos como imprimir sobre pantalla números, cadenas de caracteres y variables de cadena, utilizando como controles los signos de puntuación y **TAB**, es interesante recordar que con la misma facilidad pueden imprimirse sobre papel. Necesitaremos la impresora ZX, que tendrá que conectarse antes de enchufar, y utilizaremos **LPRINT** en lugar de **PRINT**. Este programa escribe los números impares sobre la pantalla y los pares sobre papel:

```
10  FOR J = 1 TO 20
20  IF J/2 = INT (J/2) THEN GOTO 100
30  PRINT J
40  GOTO 200
100 LPRINT J
200 NEXT J
```

Tras ejecutarlo, cambie una línea así:

```
100 LPRINT J; " ";
```

Ahora estamos haciendo que el ZX81 imprima los números pares en una línea. Observe como la impresora los guarda hasta el final del programa y entonces los imprime todos de una vez. La impresora ZX almacena el contenido de los **LPRINT** en una pequeña memoria, denominada buffer, hasta que tenga una razón para imprimirlo y pasar a la línea siguiente. Por ejemplo:

Fin de programa

Línea llena

Ultima **LPRINT** no seguida por ; o ,

TAB n inferior a la posición de impresión actual.

En este capítulo hemos aprendido ...

Instrucciones

LET para definir variables de cadena.

INPUT para detener el programa e introducir una variable de cadena.

LPRINT para imprimir sobre papel, usando la impresora ZX.

Otros

Impresión y concatenación de variables de cadena.

Comparación de variables de cadena y cadenas de caracteres mediante la instrucción **IF** y los signos =, <>, > o <.

19

Cambio de velocidad, parada y pausa

Hasta aquí, todos los programas de este libro pueden funcionar en el ZX81 y en el ZX80 (con 8K ROM). No obstante, mientras los propietarios del ZX81 habrán visto en pantalla como progresa la ejecución de los programas, los usuarios del ZX80 habrán tenido que permanecer sentados frente a una pantalla totalmente gris, esperando la salida final.

Con un ZX81 puede operar igual que con el ZX80 mediante el comando **FAST**; entonces trabajará cuatro veces más rápido que en el modo usual, **SLOW**. En cualquier caso usted tiene la posibilidad de elegir utilizando **FAST** y **SLOW** como instrucciones en sus programas. En el ejemplo que sigue podrá comparar el trabajo en modo **FAST** y en **SLOW**:

```
10  FAST
100 CLS
110 FOR J = 1 TO 40
120 LET C = EXP (LN J/3)
130 PRINT C,
140 NEXT J
200 STOP
210 SLOW
220 GOTO 100
```

Primero el programa resuelve un conjunto de 40 raíces cúbicas en modo **FAST** y más tarde, una vez que to-

das están calculadas, las muestra en pantalla (gracias a **STOP**). Si ahora pulsa **CONT**, cambiará a modo **SLOW** y repetirá el trabajo.

La elección entre **FAST** o **SLOW** es un asunto de preferencias personales. No obstante, aquí tiene una guía orientativa:

FAST es preferible:

En cálculos complicados.

En salidas por pantalla tediosas.

En programación, suponiendo que no le moleste el centelleo de la pantalla cada vez que toca una tecla.

SLOW es preferible:

En gran número de programas, especialmente los que usan gráficos.

SLOW es necesario (en otras palabras, con el ZX80 no puede hacerse):

En programas que incluyen gráficos móviles, pelotas que botan, palabras que centellean y cosas de este estilo.

A propósito, cuando usted guarda un programa queda también registrado el modo en que se encontraba el ZX81, tanto si era **FAST** como **SLOW**. Asegúrese de que se encuentra en el modo que usted desea.

Detenga su programa

He aquí una serie de hechos que hacen que se detenga la ejecución de un programa:

(1) Se ha llegado al final y el ordenador queda parado con un código Ø/n en pantalla.

(2) El operador ha pulsado **BREAK** y lo ha detenido con un código D/n visible.

(3) Hay un **STOP** en una línea de programa. En este caso el código es 9/n.

(4) Ha agotado todas las líneas disponibles en pantalla. El código es 5/n.

(5) Algún error ha ocasionado la parada. Son varios los códigos posibles.

(6) Ha llegado a una instrucción **INPUT** y está esperando que se introduzca un número o una cadena.

Este último es el más útil. Puede emplearlo para detener el programa, observar la pantalla y reiniciar cuando lo desee. Vea el programa siguiente, que muestra lo rápido que crecen las bacterias bajo condiciones favorables. Es típico que se dupliquen cada 30 minutos, aunque no tenemos en cuenta que también se quedan sin comida o mueren.

```
10  LET N = 1
20  LET T = 0
30  CLS
40  PRINT T; "HORAS QUE HAN PASADO"
50  PRINT,,"SU CULTIVO CONTIENE";N;"BACTERIAS"
60  LET T = T+0.5
70  LET N = 2*N
80  PRINT,,,, "PULSE NEWLINE PARA CONTINUAR"
90  INPUT A$
100 GOTO 30
```

Vea que podemos introducir cualquier cosa en la línea 90, aunque la cadena vacía (simplemente tecleando **NEWLINE**) ya basta para reanudar.

Ramificación y protección de programas

Podemos recurrir a una técnica similar para dar al usuario la posibilidad de dirigirse a diferentes partes del programa:

```
100 PRINT "TECLEE SI O NO"
110 INPUT A$
```

```
12Ø IF A$ = "SI" THEN GOTO 2ØØ
14Ø PRINT "USTED HA TECLEADO NO"
15Ø STOP
2ØØ PRINT "USTED HA TECLEADO SI"
```

Ejecute el programa y siga las instrucciones introduciendo SI o NO obedientemente. Luego haga una travesura y teclee PATO DONALD. El programa afirmará resueltamente: "USTED HA TECLEADO NO".

Una advertencia. Este mundo está lleno de listillos dispuestos a tratar de que el ordenador parezca tonto. También hay que tener cuidado con los novatos, aunque éstos suelen escarmentar cuando lo único que consiguen es un mensaje de error tras otro y tienen que empezar de nuevo cada vez. Todo programador debe responsabilizarse de inmunizar sus programas contra bromistas y gamberros en la medida de lo posible. Es algo difícil con una memoria de 1K, pero al menos recuerde este principio para más adelante.

Corrijamos el último programa añadiendo:

```
13Ø IF A$ <> "NO" THEN GOTO 11Ø
```

Ahora está mucho mejor. Al final del capítulo tiene dos diagramas de flujo que dejan claro lo que sucede en cada versión. Y ahora le toca a usted.

<u>Ejercicio 19.1</u> Elegir números

Escriba un programa, a prueba de graciosos, que pida un número entero entre 1 y 1ØØ, y lo imprima junto a su cuadrado. Incluya instrucciones que garanticen que realmente es entero y que está comprendido entre 1 y 1ØØ. Existe un input erróneo contra el que usted aún no puede defenderse. ¿Cuál es?.

Pausas en un programa

El ZX81 dispone de una instrucción para realizar pausas. Vea como trabaja:

```
10  PRINT "¿CUANTO?"
20  PRINT "TECLEE EL NUMERO DE SEGUNDOS"
30  INPUT S
40  PRINT S;"EMPIEZA LA SEGUNDA PAUSA"
50  PAUSE S*50
60  PRINT "HA TERMINADO"
```

La instrucción **PAUSE** n genera una pausa equivalente a n ciclos de TV (50 por segundo en el Reino Unido). Trabaja en los modos **SLOW** o **FAST**, pero en **FAST** o bien con el ZX80 el manual recomienda que a continuación de una **PAUSE** sitúe la siguiente instrucción para no perder el programa:

Número de línea **POKE** 16437,255

A mí no se me ha presentado nunca este problema con **PAUSE** en modo **FAST**, pero usted queda advertido. No se pueden hacer pausas superiores a 32767 ciclos de TV (unos 11 minutos), y si n es superior a ese número la pausa se alargará indefinidamente. Por otra parte, si pulsa cualquier tecla durante una pausa el programa reinicia inmediatamente; por tanto esto constituye otro modo de detener un programa para leer la pantalla.

```
200 PRINT "PULSE CUALQUIER TECLA PARA CONTINUAR"
210 PAUSE 40000
220 PRINT "VUELTA AL TRABAJO"
230 GOTO 220
```

En este capítulo hemos aprendido ...

<u>Comandos</u>

FAST para situar al ZX81 en modo **FAST**.

SLOW para devolver al ZX81 al modo **SLOW**.

<u>Instrucciones</u>

FAST y **SLOW** igual que antes.

PAUSE n para hacer una pausa.

Otros

Cuándo se utilizan los modos **FAST** y **SLOW**.

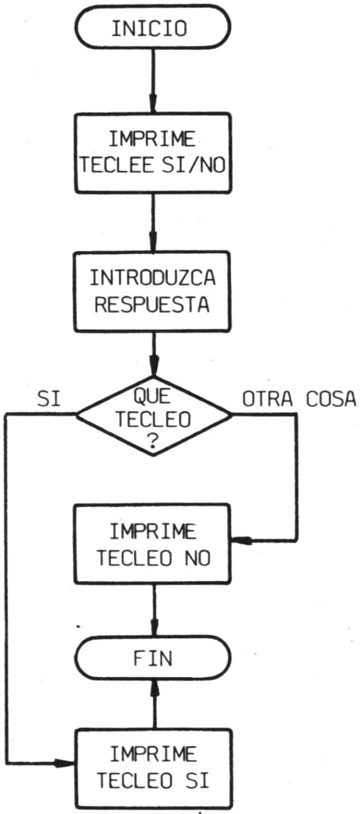
INPUT o **PAUSE** para detenciones temporales del programa.

Ramificar el programa bajo el control de una cadena a introducir por el usuario.

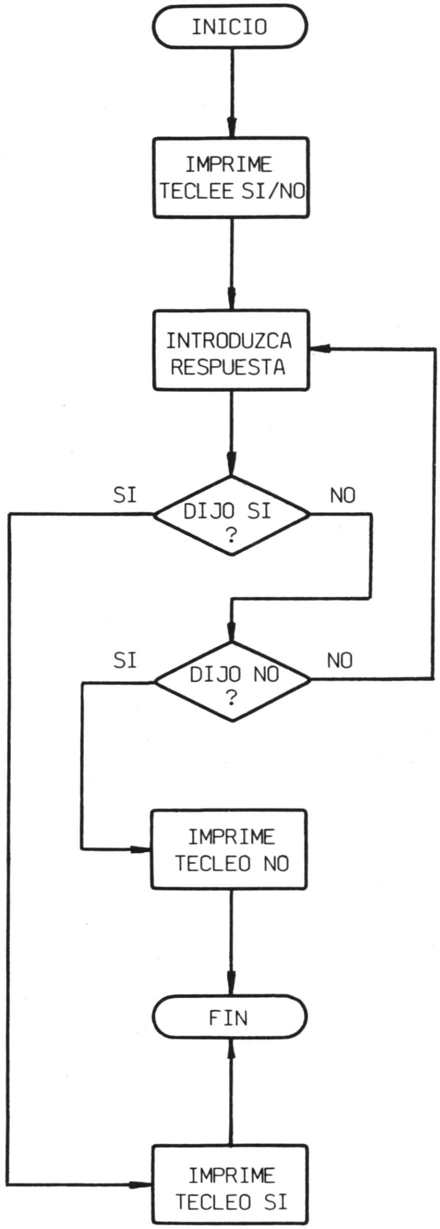
Protección de programas.

*****

Programa SI/NO (1) Versión vulnerable.



Programa SI/NO (2) Versión a prueba de graciosos.



20

Un negocio incierto

Números aleatorios

Existe un generador de números aleatorios muy sencillo que todos nosotros hemos utilizado alguna vez: el dado. Por supuesto, él también obedece ciertas normas. Sólo puede dar números del 1 al 6. Salvo que esté trucado o sea defectuoso, cada uno de los números tiene la misma probabilidad de salir. Finalmente, tratándose de un simple trozo de madera o plástico, no se ve afectado por nada que haya ocurrido antes. Estas pueden ser también las normas generales para los números aleatorios:

(1) Un número aleatorio es uno extraído de un conjunto dado de números.

(2) Cada número del conjunto tiene la misma probabilidad de ser extraído.

(3) La extracción no se ve afectada en ningún sentido por extracciones anteriores.

El ZX81 dispone de una función que genera números aleatorios: **RND**. Probémosla:

```
100 FOR J = 1 TO 20
110 PRINT RND
120 NEXT J
```

Ejecútelo varias veces. ¿Cuál es el resultado?. Conjuntos de 2Ø números, mayores que Ø y menores que 1 y que parecen realmente aleatorios. De hecho son números seudoaleatorios. Cada uno de ellos se calcula ingeniosamente a partir del anterior, con lo que las reglas (1) y (2) se respetan. No obstante, siempre empiezan con el mismo número matriz; cada vez que se conecta el ZX81 la serie se repite (puede comprobarlo si lo desea). Jugar con un dado que dé una serie fija de resultados debemos reconocer que conduce a un final que puede predecirse. Afortunadamente el ZX81 también tiene una instrucción que establece un primer número "matriz" aleatorio:

```
1Ø RAND
```

Ahora obtendrá un conjunto de números aleatorios diferente cada vez que conecte y ejecute.

Si queremos que el ZX81 actúe como un dado, ¿cómo vamos a convertir los valores de **RND** en enteros del 1 al 6?. Vea esto:

Conjunto de números generado por	Menor	Mayor
RND	Ø.ØØØ...	Ø.999...
RND * 6	Ø.ØØØ...	5.999...
RND * 6 + 1	1.ØØØ...	6.999...
INT (**RND** * 6 + 1)	1	6

Por tanto, cambie la línea 11Ø del programa, así:

```
11Ø PRINT INT (RND * 6 + 1)
```

Ahora sí que parece que estemos tirando un dado. Si tomamos **RND**, lo multiplicamos por un número y le sumamos otro, podemos convertirlo en otro número situado dentro de los límites que deseemos.

Ejercicio 20.1 Ruleta

Escriba un programa para representar en pantalla los números de la ruleta, que varían de Ø a 36. Verifique que Ø y 36 aparecen realmente.

Al final del capítulo 11 vimos varias instrucciones que necesitaban números para operar. A menudo pueden utilizarse con números aleatorios, como este programa para crear constelaciones:

```
1Ø  FOR J = Ø TO 21
2Ø  PRINT TAB RND*31;"*"
3Ø  NEXT J
4Ø  PAUSE 5Ø
5Ø  CLS
6Ø  GOTO 1Ø
```

Recuerde que no necesitamos hacer nada con **RND***31 en la línea 2Ø, puesto que será automáticamente redondeado hasta el próximo entero entre Ø y 31.

De la misma manera podemos usar los números aleatorios para definir el tamaño de un bucle **FOR/NEXT**, aunque en este caso no se produce redondeo y es aconsejable convertir los números aleatorios en enteros. Practíquelo:

Ejercicio 20.2 Rectángulo aleatorio

Escriba un programa que se valga de bucles **FOR/NEXT** anidados para dibujar un rectángulo cuyas dimensiones sean aleatorias (su longitud y su altura deben estar comprendidas entre 1 y 15).

Ramificación aleatoria

Hemos aprendido a detener la ejecución de un programa y que éste ofrezca al usuario la posibilidad de elegir entre dos o más caminos para continuar. Si recurrimos a **RND** podemos sustituir la elección voluntaria por una aleatoria. Aquí tiene un ejemplo:

```
10  PRINT "USTED REGRESA A CASA"'''
20  IF RND < Ø.5 THEN GOTO 1ØØ
3Ø  FOR J = 1 TO 15
4Ø  PRINT TAB J; " ▚▚▒▒▚▚ "
                (GRAPHICS/SHIFT YYAHYY)
5Ø  NEXT J
6Ø  PRINT,, "USTED TOMO EL CAMINO BONITO"
7Ø  GOTO 2ØØ
1ØØ PRINT,, "ATAJO"
2ØØ PRINT,, "HA LLEGADO A CASA"
```

Otra forma de ramificación aleatoria consiste en combinar **GOTO** con un número aleatorio. Aquí tiene un programa que saca bloques gráficos de una bolsa que contiene el mismo número de bloques negros, grises y arlequinados.

```
5Ø  FOR J = Ø TO 5
6Ø  LET X = 1ØØ * INT (RND*3+1)
7Ø  GOTO X
1ØØ FOR K = 1 TO 3
11Ø PRINT TAB 5*J; " ███ " (3 GRAPHICS/SPACE)
12Ø NEXT K
13Ø NEXT J
14Ø STOP
2ØØ FOR K = 1 TO 3
21Ø PRINT TAB 5*J; "▒▒▒" (3 GRAPHICS/SHIFT H)
22Ø NEXT K
23Ø NEXT J
24Ø STOP
3ØØ FOR K = 1 TO 3
31Ø PRINT TAB 5*J; " ▚▚▚ " (3 GRAPHICS/SHIFT Y)
32Ø NEXT K
33Ø NEXT  J
```

Observe que la última parte del programa se repite 3 veces en las líneas 1ØØ, 2ØØ y 3ØØ. Es un mal programa que despilfarra la memoria. En el próximo capítulo trataremos de mejorarlo.

En este capítulo hemos aprendido ...

Instrucciones

RAND para obtener un número-matriz aleatorio para el cálculo de números aleatorios.

Funciones

RND, conduce a un número seudoaleatorio situado entre Ø y 1.

Otros

Empleo de números aleatorios.

Ramificaciones aleatorias en un programa.

21

Ve, pero vuelve pronto

Subrutinas

La instrucción **GOSUB** n es similar a **GOTO** n y resulta extremadamente útil. Se encarga de decir esto al ZX81:

(1) Ve a la línea n del programa.

(2) Haz lo que se indica allí.

(3) Vuelve al punto de partida.

Aquí tiene una demostración sencilla:

```
100 PRINT "SUBRUTINA DEMOSTRACION"
110 PRINT,,"PUNTO DE PARTIDA PARA SUBR 1000"
120 GOSUB 1000
130 PRINT "DE VUELTA"
140 PRINT,,"HACIA LA SUBR 2000"
150 GOSUB 2000
160 PRINT "DE VUELTA OTRA VEZ"
```

Ejecútelo y vea qué pasa. Obedece las líneas 100 a 120, se va a la línea 1000, no la encuentra y se para. Tenemos que escribir las subrutinas:

```
1000 PRINT TAB 5; "ESTA ES LA SUBR 1000"
2000 PRINT TAB 5; "ESTAMOS EN LA SUBR 2000"
```

Aún falta algo. Fue a la línea 1ØØØ como estaba previsto, no regresó, continuó hacia la línea 2ØØØ y se detuvo. Necesitamos una instrucción que ordene el regreso desde cada subrutina: **RETURN**. También pondremos un **STOP** que actúe de barrera entre el programa principal y las subrutinas:

```
900  STOP
1000 PRINT TAB 5;"ESTA ES LA SUBR 1000"
1010 RETURN
2000 PRINT TAB 5; "ESTAMOS EN LA SUBR 2000"
2010 RETURN
```

Ejecute el programa y asegúrese de que funciona. Es útil escribir las líneas del programa en el orden de ejecución:

1ØØ, 11Ø, 12Ø, 1ØØØ, 1Ø1Ø, 13Ø, 14Ø

15Ø, 2ØØØ, 2Ø1Ø, 16Ø, 9ØØ

Ahora que ya sabemos algo sobre subrutinas podemos dar algunas normas:

(1) **GOSUB** n dirige al ZX81 directamente a la línea n, o a la siguiente si n no existe. El valor n puede consistir en un número, una variable o una expresión.

(2) El ZX81 ejecuta la subrutina igual que si fuese una parte del programa principal.

(3) La subrutina tiene que finalizar con una instrucción **RETURN**, que envía al ZX81 a la línea siguiente al **GOSUB** n de partida.

(4) Se puede pasar de una subrutina a otra, dando por supuesto que usted tiene una idea clara de lo que está haciendo.

(5) Con frecuencia es útil emplear **GOSUB** condicionales en un programa:

Si (**IF**) se cumple tal condición... entonces vete a la línea n (**THEN GOSUB** n).

(6) Sitúe todas las subrutinas al final del programa y ponga una instrucción **STOP** entre ellas y el programa principal para evitar que sean indebidamente ejecutadas.

A veces resulta conveniente incluir un **GOSUB** en un bucle. Aquí tiene una nueva versión del programa de extracción de bloques del capítulo anterior. Es mucho más corto debido al empleo de **GOSUB**.

```
10  FOR J = 0 TO 5
20  LET X = 100 * INT (RND * 3 + 1)
30  FOR K = 1 TO 3
40  GOSUB X
50  NEXT K
60  NEXT J
90  STOP
100 PRINT TAB 5 * J;"███ " (3 GRAPHICS/SPACE)
110 RETURN
200 PRINT TAB 5 * J;"▒▒▒ " (3 GRAPHICS/SHIFT H)
210 RETURN
300 PRINT TAB 5 * J;"▚▚▚ " (3 GRAPHICS/SHIFT Y)
310 RETURN
```

¿Cuándo deben utilizarse las subrutinas?

Como hemos visto, es aconsejable recurrir al empleo de subrutinas cuando tenemos que abandonar el programa principal en varios puntos para realizar la misma operación cada vez. Las subrutinas ahorran memoria al ordenador y tiempo y esfuerzo al programador.

Otra buena razón para usar subrutinas es hacer más inteligibles los programas largos. Podemos fraccionarlos en dos tipos más cortos:

Un programa principal, que puede ser corto.

Un conjunto de subrutinas, etiquetadas adecuadamente con instrucciones **REM**.

También es de gran ayuda mantener listados con todos los números y títulos de las subrutinas y de los

nombres de variable que hayamos incluido en nuestros programas.

Finalmente, a veces se escriben fragmentos de programas que pueden utilizarse más tarde en otros lugares. Esto resulta mucho más sencillo si están escritos en forma de subrutinas, susceptibles de ser transferidas en bloque a otros programas.

Ejercicio 21.1 Volumen de un depósito

En este ejercicio supondremos que los depósitos de agua son cúbicos o cilíndricos. Escriba un programa que permita al usuario elegir una de estas dos formas (rechazando las demás), introducir las dimensiones y calcular el volumen de acuerdo con las siguientes fórmulas:

Volumen del cubo = lado^3

Volumen del cilindro = $\text{altura} \times \pi \times (\text{diámetro}/2)^2$

En este capítulo hemos aprendido ...

Instrucciones

GOSUB n para dirigir al ZX81 a una subrutina que empieza en la línea n.

RETURN al final de una subrutina sirve para hacer que el ZX81 regrese y continúe ejecutando el programa principal.

STOP para establecer una separación entre las subrutinas y el programa principal.

22

Aceleremos la entrada

Hasta ahora, para introducir números o cadenas hemos tenido que detener el programa, teclearlas y pulsar **NEWLINE**. Una nueva función, **INKEY$**, nos permite hacer eso mismo más rápida y cómodamente, aunque no sin algunas limitaciones (por ejemplo, necesita que el ZX81 trabaje en modo **SLOW**).

Cuando el ZX81 se encuentra con **INKEY$**, inmediatamente verifica la situación de todo el teclado. Si hay alguna tecla presionada, con o sin **SHIFT**, el carácter correspondiente se asigna a una variable de cadena de un solo carácter, etiquetada como **INKEY$**. Pruébelo:

```
1ØØ PRINT INKEY$;
```

¡Ciertamente se trata de una laboriosa manera de escribir!. Usted ha tenido que mantener presionada la tecla **NEWLINE**, que a su vez nos ha devuelto un ?. Ahora vamos a incluir **INKEY$** en un bucle para que podamos retirar el dedo de **NEWLINE**.

```
11Ø GOTO 1ØØ
```

Diviértase un poco tocando letras y letras, pero recuerde que por cada carácter que aparezca en pantalla, el ZX81 ha explorado todas las teclas y asignado un nuevo carácter a **INKEY$**. A propósito, usted no pue-

de pulsar **SPACE** o £: el ZX81 las tomaría como si fuesen **BREAK**.

Supongo que se ha dado cuenta de que **INKEY$** es algo realmente efímero. Siempre que usted presione **INKEY$** en un programa, se genera uno nuevo. Por tanto tendremos que recurrir a algunos trucos para utilizarlo.

Ramificación de programas

Aquí tiene un ejemplo rápido y sencillo de programa en el que el usuario puede elegir el camino tras una ramificación. Es similar al del capítulo 19:

```
10  PRINT "¿SIGUE O PARA?"
20  PRINT,,"PULSE S O P"
30  IF INKEY$ = "S" THEN GOTO 200
40  IF INKEY$ = "P" THEN GOTO 100
50  GOTO  30
100 PRINT,,"HA PARADO"
110 STOP
200 PRINT,,"CONTINUA"
```

¡Casi perfecto!. Pulse cualquier tecla que desee y el ZX81 ejecutará las instrucciones 30, 40 y 50 hasta que sean presionadas S o P.

Una grabación permanente de INKEY$

En el programa anterior se empleaba **INKEY$** y después se olvidaba. No obstante, hay ocasiones en que necesitamos una grabación permanente del mismo, como ésta por ejemplo:

```
10  PRINT "PULSE CUALQUIER TECLA"
100 IF INKEY$ <> " " THEN GOTO 100
110 IF INKEY$ = " " THEN GOTO 110
120 LET A$ = INKEY$
130 PRINT "SU INKEY$ ERA";A$
```

Esto requiere una explicación. La línea 100 retiene el programa mientras no se presione alguna tecla, dán-

dole a usted la oportunidad de dejar de oprimir la tecla **NEWLINE**. Luego, la línea 11Ø lo para mientras se pulsa una tecla y finalmente la línea 12Ø asigna el carácter **INKEY$** a A$. Ejecute el programa y teclee los comandos:

```
PRINT INKEY$   (Ha desaparecido)

PRINT A$       (Permanece todavía)
```

Añadiendo algunas cosas podemos utilizar **INKEY$** para introducir cadenas de cualquier longitud, previamente definida.

```
1Ø  PRINT "TECLEE UNA PALABRA DE TRES LETRAS"
2Ø  LET A$ = ""
1ØØ FOR J = 1 TO 3
11Ø IF INKEY$ <> "" THEN GOTO 11Ø
12Ø IF INKEY$ = "" THEN GOTO 12Ø
13Ø LET A$ = A$ + INKEY$
14Ø NEXT J
15Ø PRINT "SU PALABRA ERA";A$
```

Se puede jugar con **INKEY$** para introducir cadenas de longitud indeterminada, pero ello no resulta más ventajoso que usar **INPUT**.

¿Y respecto a los números?

Si en el programa anterior se introduce 123, el resultado parece un número, pero en realidad es la cadena "123" razón por la cual no puede utilizarse en operaciones matemáticas. Por fortuna, el ZX81 dispone de una función que convierte cadenas en números. Modifique y amplíe el programa, así:

```
15Ø PRINT,, "CADENA A$","VAL A$"
16Ø PRINT A$, VAL A$
17Ø GOTO 1Ø
```

Pruebe utilizando cadenas de todas clases, incluso algunas como éstas: "123", "4.5", "6 + 7", "89A".

Supongo que habrá descubierto ya las normas de trabajo de **VAL**:

(1) Si una cadena está totalmente formada por caracteres que pueden utilizarse en una expresión aritmética, **VAL** calculará el resultado de la expresión. Estos caracteres son:

Números
Nombres de variable previamente definidas
Operadores
Punto
Funciones
Paréntesis

(2) Cualquier otro tipo de carácter ocasionará una parada con código de error C/n o 2/n.

(3) Se puede grabar una cadena **VAL** asignándola a una variable numérica.

LET A = **VAL** A$

El ZX81 también tiene otra función que hace exactamente lo contrario que **VAL**. Se conoce como **STR$**.

STR$ número = "número"

STR$ 567 = "567"

Por ahora nos contentamos con haber mencionado la función **STR$**. Ya la estudiaremos más tarde.

Aquí tiene un conocido programa para que usted lo escriba utilizando **INKEY$** y **VAL**.

<u>Ejercicio 22.1</u> Adivine un número

Escriba un programa que genere un número aleatorio entre 1Ø y 99. Pida al usuario que trate de adivinarlo y luego dígale si el número que ha tecleado es demasiado alto, demasiado bajo o correcto. Si sólo dispone de 1K de memoria se verá obligado a limitar a 8 el número de tentativas.

En este capítulo hemos aprendido ...

Funciones

INKEY$ para introducir una cadena de un solo carácter sin detener la ejecución.

VAL para convertir una cadena en un número.

STR$ para convertir un número en una cadena.

23

Gráficos

En el capítulo 17 ya hicimos gráficos sencillos mediante la instrucción **PRINT** y los bloques gráficos entrecomillados. Cada bloque o elemento gráfico constaba de cuatro cuadraditos (pixels) que podían ser negros, blancos o grises.

Trazado de puntos

Podemos emplear la instrucción **PLOT** X,Y para situar un pixel negro en cualquier punto de la pantalla. Estudie este programa de demostración:

```
20  PRINT "DEMOSTRACION DE PLOT X,Y"
30  PRINT,,"Ø A 63 HORIZONTALMENTE ES X"
40  PRINT,,"Ø A 43 VERTICALMENTE ES Y"
50  PRINT,,"INTRODUZCA X (Ø A 63) X =  ";
60  INPUT X
70  PRINT X,,, "AHORA Y (Ø A 43)"
80  INPUT Y
90  CLS
100 PLOT  X,Y
110 PRINT X;",";Y
120 INPUT A$
130 CLS
140 GOTO  50
```

Si ejecuta el programa verá que se explica por sí mismo. Observe que la posición **PRINT** de la línea 11Ø está inmediatamente después de la posición **PLOT** de la línea 1ØØ.

La instrucción **CLS** de la línea 13Ø es algo así como un borrador que recorre toda la pizarra para borrar un solo punto. Podemos hacer lo mismo de un modo menos rudimentario utilizando **UNPLOT**, el inverso de **PLOT**.

```
13Ø UNPLOT X,Y
```

Observe otra vez la posición de **PRINT**, situada inmediatamente después de la posición de **UNPLOT**.

Trazado de líneas

Una mancha negra aislada no es algo demasiado útil, pero vea qué pasa cuando la introducimos en un bucle:

```
1Ø  FOR J = Ø TO 63
2Ø  PLOT J,Ø
9Ø  NEXT J
```

Es el principio de un marco. Ahora necesitamos una línea horizontal en la parte alta de la pantalla. ¿Se atreve a hacerlo? ¡Pues claro que sí!.

```
3Ø  PLOT J,43
```

El resto lo dejo para usted.

Ejercicio 23.1 Líneas verticales

Añada cuatro instrucciones más al programa anterior para dibujar las dos líneas verticales que faltan para completar el marco. Por cierto, aquí se plantean problemas con las memorias de 1K.

Las líneas oblicuas ya son otra cosa. En ningún caso podrán quedarnos tan bien. De todos modos, veamos qué es lo que podemos hacer:

```
10  FOR  J = 0 TO 43
20  PLOT J,0
30  PLOT J,43
40  PLOT 0,J
50  PLOT 43,J
60  PLOT J,J
70  PLOT J,43 - J
100 NEXT J
```

Introduzca las sentencias 20 a 70 una a una y ejecute cada vez para ver qué línea es dibujada en pantalla por cada instrucción.

Si lo desea puede utilizar **PLOT** en bucles anidados para ennegrecer franjas completas de la pantalla, aunque este procedimiento resulta un poco lento.

```
10  FOR  J = 0 TO 63
20  FOR  K = 0 TO 41
30  PLOT J,K
40  NEXT K
50  NEXT J
```

Podemos borrar la pantalla por el método usual:

```
60 CLS
```

No obstante, si reducimos el tamaño del rectángulo para liberar memoria, borraremos más cómodamente utilizando el método del "queso danés":

```
10  LET  K = 0
20  FOR  J = 0 TO 43
30  FOR  K = 0 TO K
40  PLOT J,K
50  NEXT K
60  NEXT J
70  LET  X = RND * 43
80  UNPLOT X,RND * (X + 1)
90  GOTO 70
```

Combinemos PRINT con gráficos

Como ya sabemos, la posición **PRINT** sigue inmediatamente al último punto **PLOT** o **UNPLOT**, lo que en ocasiones puede ser un inconveniente. Por suerte, el ZX81 nos permite imprimir a voluntad en cualquier punto mediante la instrucción:

PRINT AT número de línea, número de columna, cadena o número.

Los números de línea van desde Ø en la parte superior de la pantalla, hasta 21 en la inferior. Los números de columna son los mismos que para **TAB**, del Ø al 31. Aquí tiene un juego para que se familiarice con **PRINT AT**:

```
10  PRINT TAB 7;"DEMOSTRACION DE PRINT AT"
20  PAUSE 200
100 CLS
110 PRINT "PONGA UN DEDO EN ESTOS PUNTOS"
120 PAUSE 200
130 LET L = INT (RND * 22)
140 LET C = INT (RND * 32)
150 CLS
160 PRINT "PRINT AT "; L; ","; C
170 PAUSE 400
180 PRINT AT L, C; "*"
190 GOTO 120
```

Recuerde que cualquier cosa que se imprima se hará inmediatamente después del punto **PRINT AT**, de acuerdo con las normas de puntuación usuales. Si desea retroceder tendrá que utilizar otro **PRINT AT**.

Otra aplicación de **PRINT AT** es el borrado de puntos concretos de la pantalla. Todo lo que tiene que hacer es imprimir blancos sobre los puntos que desea borrar.

```
100 FOR J = 0 TO 21
110 PRINT AT J,J;J
120 NEXT J
200 PRINT AT 0,4;"BORREMOS LOS NUMEROS IMPARES"
```

```
210 FOR J = 1 TO 21 STEP 2
220 PRINT AT J,J;"   "
230 NEXT J
```

Aquí tiene unos ejercicios sobre **PLOT** y **PRINT AT.**

Ejercicio 23.2 Tarjeta de visita

Escriba un programa que imprima una tarjeta de visita negra en el centro de la pantalla, con su nombre y dirección en letras blancas.

Ejercicio 23.3 Subrutina "continuamos"

Se trata de una subrutina muy útil para detener el programa hasta que presionemos **NEWLINE.** Haga que se imprima el mensaje "PULSE NEWLINE" en la parte inferior derecha de la pantalla, haga una pausa con **INPUT,** borre la línea inferior y ponga un **RETURN.**

Gráficos con la impresora ZX

Los gráficos sencillos del capítulo 17, en los que la impresión se realiza línea a línea, pueden ser obtenidos sobre papel simplemente cambiando las instrucciones **PRINT** por **LPRINT.** Por el contrario, **LPRINT AT** no funciona; viene a ser más o menos igual que **LPRINT TAB.** Si medita un poco sobre ello se dará cuenta de que **PRINT AT** puede pedirle al ZX81 que retroceda sobre una línea o sobre líneas anteriores, y la impresora ZX no puede hacer cosas así. Tampoco **PLOT** sirve con ella. ¿Entonces qué debemos hacer para registrar nuestros bonitos gráficos?.

La respuesta se halla en el capítulo 15: sencillamente utilice la palabra clave **COPY,** ya sea como instrucción, ya como comando, y la impresora ZX hará una copia fiel del contenido de la pantalla.

En este capítulo hemos aprendido ...

Instrucciones

PLOT X,Y para situar un pixel negro en la posición de coordenadas X,Y.

UNPLOT X,Y borra un pixel de coordenadas X,Y.

PRINT AT número de línea, número de columna; se usa para imprimir algo en cierta posición, con independencia de lo que se haya impreso anteriomente.

COPY para obtener un registro sobre papel del contenido de la pantalla.

Otros

Bucles con **PLOT** para dibujar líneas y bloques en la pantalla.

PRINT AT línea, columna;" " para borrar zonas de la pantalla.

24

Juguemos con las cadenas

Hay una teoría que dice que un grupo de chimpancés llegarían a escribir las obras completas de Shakespeare, con la condición de que dispusieran de tiempo y de papel ilimitados. Probemos:

```
10  RAND
90  CLS
100 FOR  J = 1 TO 80
200 FOR  K = 1 TO INT (RND * 8 + 1)
210 LET  A = INT (RND * 26 + 38)
220 PRINT CHR$ A;
300 NEXT K
310 IF RND < .07 THEN PRINT ".";
350 PRINT " ";
400 NEXT J
500 PRINT
510 PRINT,, "PULSE NEWLINE"
520 INPUT A$
530 GOTO 90
```

Supongo que esa teoría es cierta, pero ¡hace falta paciencia para comprobarlo!. Las líneas interesantes del programa son la 210, que genera un número aleatorio entre 38 y 63, y la 220 que imprime una nueva función: **CHR$**. En este capítulo la veremos con amplitud.

CHR$ A es el carácter cuyo código es el número A. Si mira la página 182 del manual de instrucciones, verá que los números 38 a 63 son los códigos de las letras del alfabeto.

Podemos emplear **CHR$** para ver cualquier carácter de todo el repertorio del ZX81, que son 255 en total.

```
10  LET K = 0
20  FOR J = 1 TO 8
30  FOR K = K TO K+7
40  PRINT CHR$ K;"  ";
50  NEXT K
60  PRINT,,
70  NEXT J
80  PRINT,,"PULSE NEWLINE"
90  INPUT A$
100 CLS
110 GOTO 20
```

Podrá ver un conjunto de bloques gráficos, números, símbolos, letras, palabras clave, funciones y caracteres inversos. La segunda página consiste mayormente en interrogantes; se trata de caracteres sin utilidad o comandos como **NEWLINE** que no salen por pantalla.

Hay dos funciones más que están relacionadas con las cadenas y que resultan sumamente útiles: **CODE** y **LEN**. Este programa esclarece lo que hacen:

```
10  PRINT "INTRODUZCA ALGUNAS PALABRAS"
20  PRINT "W$";TAB 10;"CODE W$";TAB 20;"LEN W$"
30  INPUT W$
40  PRINT,, W$;TAB 10;CODE W$;TAB 20;LEN W$
50  GOTO 30
```

Ejecute el programa e introduzca palabras tales como ASTRO, ARCO, A, BELLO, BIEN, B. Introduzca también espacios e incluso una cadena vacía. A estas alturas habrá descubierto ya que **CODE** y una cadena dan "el código numérico que corresponde al primer carácter de esa cadena". Y **LEN** de una cadena es igual "al número de caracteres, incluidos los espacios, de que consta la cadena"; en otras palabras, su longitud.

Fragmentación de cadenas

El ZX81 dispone de un sencillo pero eficaz procedimiento para trocear cadenas. Tan pronto como se teclea una cadena o una variable de cadena, cada carácter va siendo numerado con un número correlativo que empieza en 1 y acaba en el número que corresponde a **LEN**. Por ejemplo:

LET Z$ = "HILO" **LEN** Z$ = 4

Z$(1) = "H" Z$(2) = "I" Z$(3) = "L" Z$(4) = "O"

Podemos 'cortar' los caracteres que queramos de una cadena utilizando la función:

cadena (m **TO** N)

Haga la prueba con este programa:

```
10  PRINT "DEPORTISTA"
20  LET A$ = "DEPORTISTA"
100 PRINT "INTRODUZCA DOS NUMEROS DEL 1 AL 10"
110 PAUSE 300
120 CLS
130 INPUT M
140 INPUT N
150 PRINT,,"DEPORTISTA(";M;"TO";N;") =";A$(M TO N)
160 GOTO 130
```

Si introduce varios pares de números verá que el primero no debe ser inferior a 1, y el segundo no puede ser mayor que 10 (**LEN** "DEPORTISTA" = 10).

Si lo desea puede cortar una parte de una cadena y asignarla a otra variable de cadena, para usarla más adelante. Ejecute el programa anterior y teclee estos comandos:

```
LET B$ = A$ (2 TO 8)
PRINT A$,B$
```

Es posible que usted sólo necesite un carácter de la cadena original. En este caso puede prescindir del

TO. Para verlo, teclee los siguientes comandos y otros similares:

```
PRINT A$(1)     PRINT A$(2)     PRINT A$(9)
```

Nuevamente, el límite inferior es 1, y el superior es **LEN** A$. Utilice ahora este método para seleccionar e imprimir palabras de diversas maneras:

```
1Ø  PRINT "INTRODUZCA UNA PALABRA"
2Ø  INPUT W$
3Ø  CLS
1ØØ FOR J = 1 TO LEN W$
11Ø PRINT W$(J);" ";
12Ø NEXT J
```

Hemos respetado el orden original, pero ahora modifique la línea 1ØØ así:

```
1ØØ FOR J = LEN W$ TO 1 STEP -1
```

Vea que es la misma palabra pero en orden inverso. También podemos servirnos del hecho de que el **CODE** de una letra invertida es mayor en 128 que el **CODE** de la letra original:

```
11Ø PRINT TAB 1;CHR$(CODE W$(J)+128)
```

¡Perdón, ahora las hemos puesto en posición descendente!. Añada estas líneas para hacer que cada letra se vaya a su sitio:

```
2ØØ FOR  J = 1 TO LEN W$
21Ø FOR  K = Ø TO J-1
22Ø PRINT AT LEN W$-J+K,K+1;CHR$(CODE W$(J)+128)
23Ø PRINT AT LEN W$-J+K-1,K;" "
24Ø NEXT K
25Ø NEXT J
```

Aquí tiene un ejercicio para que practique y pueda cortar cadenas a su gusto.

Ejercicio 24.1 CAN

Vamos a llamar "can" a cualquier palabra que empiece o termine con estas tres letras. Escriba un programa que solicite palabras, las verifique, liste aquellas que son "can" y rechace las que no lo son.

En este capítulo hemos aprendido ...

Funciones

CHR$ n, que es igual al carácter cuyo código numérico es n.

CODE s es igual al código numérico del primer carácter de la cadena s.

LEN s es el número de caracteres contenidos en la cadena s.

s (m **TO** n) es un fragmento de la cadena s que empieza en el carácter m y acaba en el n.

s (n) es el carácter número n de la cadena s.

25

Tablas

Variables ficticias

Hemos aprendido la manera de hacer toda clase de operaciones con números y con cadenas. A veces necesitamos disponer de una grabación permanente del número o de la cadena originales, para lo cual utilizamos una variable ficticia. Aunque las estudiaremos más tarde, veamos ahora un ejemplo sencillo:

```
10  LET B$ = ""
100 PRINT "TECLEE UNA PALABRA"
110 INPUT A$
120 CLS
130 PRINT "USTED HA TECLEADO ";A$
140 IF A$ = B$ THEN GOTO 300
200 PRINT "ESTO ES UN CAMBIO"
210 PRINT "ERA ";B$;" LA ULTIMA VEZ"
220 GOTO 400
300 PRINT "ABURRIDO - IGUAL QUE LA ULTIMA VEZ"
400 LET B$ = A$
410 GOTO 100
```

Tenemos un bucle con **INPUT** entre las líneas 100 y 400, y la variable A$ cambia cada vez que se ejecuta el bucle. Sin embargo, en la línea 400 asignamos A$ a la variable ficticia B$, de manera que podremos compararla con el próximo valor de A$. Por supuesto, podemos hacer lo mismo con variables numéricas.

Tablas numéricas

Ya sabemos cómo obtener una grabación permanente de un número asignándolo a un nombre de variable. Ahora suponga que deseamos grabar un conjunto de números que tienen algo en común. Por ejemplo, el número de unos, doses, treses, cuatros, cincos y seises que han salido en un juego de lanzamiento de dados. Una cosa así podemos hacerla definiendo una tabla monodimensional mediante la instrucción **DIM**:

```
10 DIM D(6)
```

Si ahora ejecutamos este programa, habremos creado seis variables:

D(1) D(2) D(3) D(4) D(5) D(6)

cada una de las cuales ha sido puesta a cero. Compruébelo tecleando comandos como **PRINT** D(3).

Una tabla debe tener un nombre de una sola letra. El número de miembros que la integran puede ser cualquiera, con la única restricción de la memoria disponible. Cada miembro o elemento se distingue de los demás por un subíndice entre paréntesis que es distinto para cada uno de ellos y que empieza por 1. Observe que D(0) no existe.

Veamos un programa en el que se tira un dado 60 veces tras una aleatorización previa.

```
20  RAND
100 FOR J = 1 TO 60
110 LET T = INT (RND*6 + 1)
200 NEXT J
```

Y ahora es cuando intervienen los subíndices. Si en una tirada sale un 5, añadiremos uno a D(5), que expresa el número de cincos que han salido.

```
120 LET D(T) = D(T) + 1
```

Si, por ejemplo, T es igual a cinco, la línea anterior equivale a:

```
LET D(5) = D(5) + 1
```

Si sucede que el próximo valor de T es 3, sumaremos 1 a D(3) y así sucesivamente. Ahora tenemos que imprimir los resultados:

```
9Ø  PRINT "ESPERE"
3ØØ PRINT "6Ø TIRADAS",,
31Ø FOR J = 1 TO 6
32Ø PRINT TAB 5;D(J);" ";J;"S",
33Ø NEXT J
```

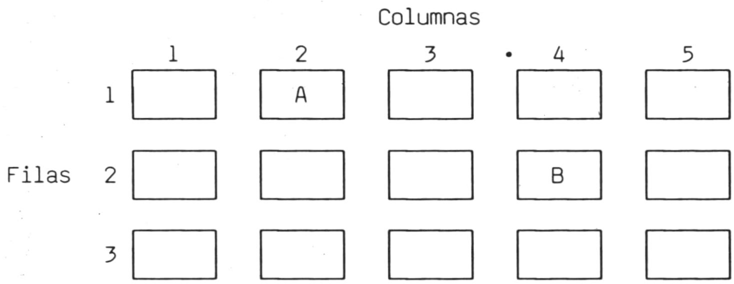
Finalmente, vamos a tener la posibilidad de detener la obtención de resultados y volver al **DIM** de la línea 1Ø para poner a cero todas las variables y empezar de nuevo:

```
4ØØ PRINT "SI QUIERE CONTINUAR PULSE N/L"
41Ø INPUT A$
42Ø GOTO 1Ø
```

Más adelante, cuando veamos los gráficos móviles, escribiremos otra vez este programa para obtener un juego de competición.

Tablas multidimensionales

Imagine que estamos tratando de alquilar quince caravanas en agosto a los veraneantes. Las tenemos dispuestas en tres filas y cinco columnas:



Podemos designar una caravana simplemente dando su fila y su columna. A título de ejemplo, la caravana A es la 1,2 (fila 1 y columna 2). La B es la 2,4.

Exactamente lo mismo hace el ZX81 cuando empleamos la instrucción **DIM**:

```
10 DIM V(3,5)
```

Esta vez al ejecutar el programa tendremos 15 variables dispuestas en una tabla de 3 x 5, como las caravanas anteriores, y con valor cero todas ellas.

V(1,1) = 0   V(1,2) = 0   etcétera.

Si ahora acordamos que V(m,n) = 0 significa que una caravana está libre y que V(m,n) = 1 quiere decir que esa caravana ya está contratada, estaremos en condiciones de escribir un programa de alquiler de caravanas:

```
20  PRINT "¿QUE CARAVANA QUIERE?"
30  PRINT ,,"¿QUE FILA (DE 1 A 3)?";
40  INPUT R
50  PRINT R
60  PRINT "¿QUE COLUMNA (DE 1 A 5)?";
70  INPUT C
80  PRINT C
90  PAUSE 200
100 IF V(R,C) = 1 THEN GOTO 200
110 PRINT ,,"LA CARAVANA (";R;",";C;") ESTA LIBRE"
120 LET V (R,C) = 1
130 PRINT ,,"TRATO HECHO"
140 PRINT ,,"EL SIGUIENTE POR FAVOR - PULSE N/L"
150 GOTO  220
200 PRINT ,,"PERDON, LA CARAVANA (";R;",";C;") NO
             ESTA LIBRE"
210 PRINT ,,"PULSE N/L PARA PROBAR OTRA"
220 INPUT A$
230 CLS
240 GOTO  20
```

Si disponemos de memoria suficiente, no estamos limitados a dos dimensiones. Cada caravana podría alqui-

larse uno cualquiera de los doce meses del año, con lo que necesitaríamos una tabla de 3 x 5 x 12. El programa empezaría así:

```
10  DIM V(3,5,12)
```

¡pero será usted quien tenga que escribir el resto!.

Aquí tiene un problema sobre tablas.

Ejercicio 25.1 Toros y vacas

Se trata de un juego muy conocido en el que hay que adivinar un número de cuatro cifras. Tras su pronóstico, usted es informado del número de cifras que acertó (toros). El esquema general es éste:

Se generan cuatro cifras aleatorias entre 1 y 6, y se las integra en una tabla de una dimensión.

Se pide al jugador que trate de adivinar el número.

Ahora se introduce el pronóstico como una variable de cadena.

Luego hay que comparar las cifras de la respuesta, una por una, con las cifras de la tabla (acuérdese de **VAL**).

Se informa al jugador de los toros conseguidos.

En este capítulo hemos aprendido ...

Instrucciones

DIM reserva espacio para una tabla de números y la carga con ceros. Por ejemplo, **DIM** A(n) define una tabla de una sola dimensión con n miembros, y **DIM** A(m,n) genera una tabla de m x n elementos.

Otros

Variables ficticias para que no se pierdan determinadas variables.

26

Tablas de cadenas

Hemos tratado las tablas de números en el último capítulo. Las tablas de cadenas son muy parecidas. Ya sabemos que una variable de cadena es equivalente a una cadena de caracteres de una sola dimensión. Por ejemplo:

```
20  LET A$ = "ALA"
```

Ejecute el programa y teclee estos comandos:

```
PRINT A$(1)      (esto da A)
PRINT A$(2)      (esto da L)
PRINT A$(3)      (esto da A)
```

Si escribimos una instrucción **DIM** como ésta:

```
10  DIM A$(5)
```

habremos reservado espacio para una cadena de cinco caracteres denominada A$, asignado todos los caracteres a espacios vacíos e insertado la cadena "ALA". Esto puede comprobarse añadiendo:

```
30  FOR J = 1 TO 3
40  FOR K = 1 TO 5
50  PRINT A$(K)
60  NEXT K
70  NEXT J
```

Hay cinco espacios disponibles, pero únicamente hemos ocupado tres. Ahora vamos a cambiar la instrucción **DIM**. Lo mejor es teclear **NEW** y empezar de nuevo.

```
10 DIM B$(7,5)
```

Esta vez hemos reservado espacio para una tabla de siete cadenas, cada una de ellas de cinco caracteres. Empecemos con éstas:

```
20 LET B$(1) = "ALA"
30 LET B$(2) = "MAR"
40 LET B$(4) = "RATON"
```

y ahora las imprimimos:

```
100 FOR J = 1 TO 7
110 PRINT B$(J);
120 NEXT J
```

Observe que en la definición de los miembros de esta tabla (líneas 20 a 40) y en su empleo (línea 110), únicamente hemos tecleado un subíndice para indicar la cadena acerca de la cual estamos hablando. El segundo subíndice se utiliza una sola vez (en la instrucción **DIM**) para fijar la longitud máxima de cada miembro de la tabla. ¿Y qué sucede si intentamos introducir una cadena más larga de lo permitido?.

```
50 LET B$(5) = "ELEFANTE"
```

El ZX81 no objeta nada, simplemente rehúsa imprimir más caracteres de los cinco primeros. Si usted necesita algo más que "ELEFA", tendrá que cambiar la instrucción **DIM**. ¡A lo mejor es por esto por lo que mi carnet de conducir está extendido a nombre de un joven llamado "NORMA"!.

Tablas de cadenas multidimensionales

Son igual de fáciles, pero requieren algo más de memoria. Teclee el comando **NEW** y luego **DIM** C$(4,3,8).

Con esto hemos reservado espacio para una tabla de 4x3 cadenas. Una vez más, el último subíndice sirve para fijar la longitud máxima de las cadenas, y sólo aparece en la instrucción **DIM.** Cuando se definen o se utilizan miembros de la tabla empleamos tan sólo los dos primeros números.

```
LET C$(2,3) = "ELEFANTE"
PRINT C$(2,3)
```

Denominación de tablas de cadenas

Una tabla puede tener un nombre de una sola letra, seguida del signo $ y de los subíndices. Un nombre como A$, por ejemplo, no puede utilizarse más que para una tabla de cadenas. Si luego introducimos una segunda **DIM** A$(m,n,...) sencillamente estamos cancelando la **DIM** original y reemplazándola por la nueva. No obstante, si lo desea puede usar todas estas variables en un programa:

A (variable numérica)

A (n,...) (tabla numérica)

A$ (variable de cadena)

A$ (n,...) (tabla de cadenas)

Eliminar miembros de una tabla de cadenas

Suponiendo que todavía no ha borrado la instrucción **LET** C$ = "ELEFANTE", teclee estos comandos:

```
PRINT C$(2,3,1)
PRINT C$(2,3,2)
PRINT C$(2,3,8)
```

Es obvio que si introduce un subíndice extra, lo que está haciendo es eliminar ese carácter en particular de la variable de cadena. Si quiere recortar más, hágalo así:

PRINT C$(2,3)(2 **TO** 7) o bien **PRINT** C$(2,3,2 **TO** 7)

Y ahora practique todo lo anterior con los siguientes ejercicios de tablas de cadenas:

Ejercicio 26.1 Calificación de un examen

Imagine que tiene seis alumnos en clase. Introduzca sus nombres en una tabla de cadenas y escriba un programa que solicite:

El nombre del ejercicio.

La calificación máxima posible.

La nota de cada alumno (use una tabla numérica).

La salida debe consistir en un título y una lista de nombres y porcentajes.

Ejercicio 26.2 Máquina tragaperras

Establezca una tabla de cadenas que contenga seis figuras de las utilizadas en las máquinas tragaperras (campana, limón, etc). Genere tres números aleatorios y úselos para llevar a pantalla tres figuras. Compruebe si hay premio (tres figuras iguales).

En este capítulo hemos aprendido ...

Instrucciones

DIM A$(m,n...) para crear tablas de cadenas mono o multidimensionales. El último subíndice (extra) fija la longitud máxima de cada elemento de la tabla.

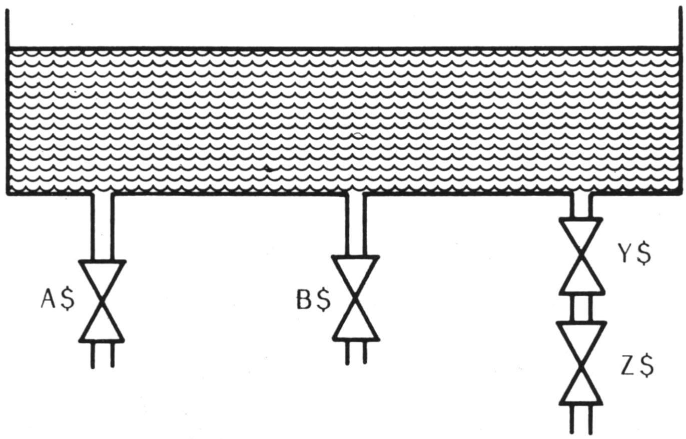
Otros

Eliminación de caracteres o fragmentos de los miembros de una tabla de cadenas.

27

Muy lógico

En el capítulo 12 ya empezamos con **IF... THEN**. Volvamos ahora sobre ello. Aquí tiene el dibujo de un depósito de agua con un curioso sistema de tuberías. Hay cuatro grifos, denominados A$, B$, Y$ y Z$.



Se trata de un problema sencillo de ingeniería. Tenemos que escribir un programa que nos avise cuándo se nos escape el agua a través de un grifo abierto. En primer lugar veremos A$:

```
10   PRINT "FIJE LA POSICION DE LOS GRIFOS",,
           "A = ABIERTO  C = CERRADO"
20   PRINT ,,"¿COMO ESTA A$?";
30   INPUT A$
40   PRINT A$
150  IF A$ = "A" THEN GOTO 1000
200  PRINT "TODO BIEN"
210  PRINT AT 21,0;"PULSE N/L PARA SEGUIR
              O BIEN S PARA PARAR"
220  INPUT X$
230  IF X$ = "S" THEN STOP
240  CLS
250  GOTO 10
1000 PRINT
1010 FOR J = 1 TO 5
1020 PRINT "DING DONG"
1030 NEXT J
1040 PRINT ,,"SE ESTA ESCAPANDO EL AGUA"
1050 GOTO 210
```

Ejecute el programa, abra y cierre A$ y asegúrese de que la alarma funciona correctamente. Vamos ahora con el grifo B$.

```
50  PRINT "¿COMO ESTA B$?";
60  INPUT B$
70  PRINT B$
```

Ahora necesitamos una línea como la 150 para comprobar si B$ está abierto, pero... un momento, creo que podemos incluir B$ en la línea 150:

```
150 IF A$ = "A" OR B$ = "A" THEN GOTO 1000
```

¿Qué, funciona?¡Pues claro que sí!. La alarma actúa si está abierto tanto A$ como B$. Vamos a ocuparnos de los otros dos grifos ahora:

```
80  PRINT "¿COMO ESTA Y$?";
90  INPUT Y$
100 PRINT Y$
110 PRINT "¿COMO ESTA Z$?";
120 INPUT Z$
130 PRINT Z$
```

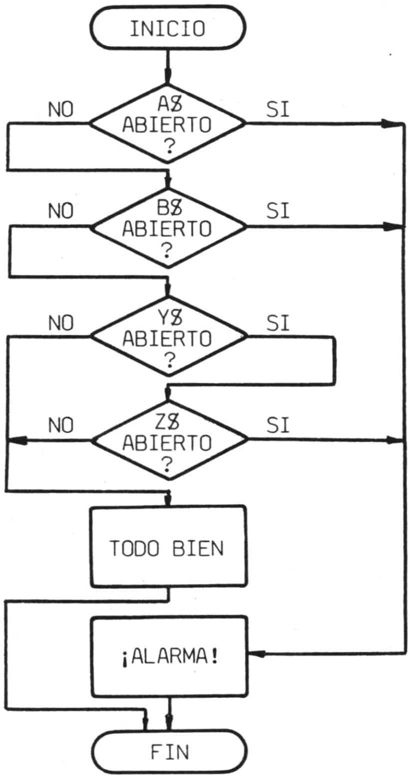
Tendremos que pensar detenidamente sobre todo esto. Si cualquiera de los dos grifos, Y$ o Z$, permanece cerrado, estaremos manteniendo el agua en el tanque. Sólo tendremos motivos de alarma si ambos están abiertos, así:

```
160 IF A$ = "A" AND Z$ = "A" THEN GOTO 1000
```

Vuelva a ejecutar el programa y abra y cierre todos los grifos para comprobar que todo va bien. Luego sustituya las líneas 15Ø y 16Ø por una sola rigurosamente lógica:

```
15Ø IF A$ = "A" OR B$ = "A" OR Y$ ="A" AND Z$ = "A"
      THEN GOTO 1ØØØ
```

que hace el mismo trabajo. Vea el diagrama de flujo de este programa:



Prioridades

Las instrucciones lógicas tan largas nos exigen que tengamos las ideas claras. Su acción depende del hecho de que el ZX81 verifica la instrucción en un orden determinado, concediendo prioridad a **AND** sobre **OR**. Igual que en las expresiones aritméticas, podemos cambiar la prioridad, o bien enfatizarla, utilizando paréntesis. Por ejemplo, esta línea:

```
15Ø IF A$ = "A" OR B$ = "A" OR (Y$ = "A" AND
      Z$ = "A") THEN GOTO 1ØØØ
```

tiene exactamente el mismo efecto que la anterior, pero resulta más inteligible.

Este es el momento de decir que el ZX81 dispone de un **NOT** lógico, aunque parezca superfluo dado que:

```
IF NOT A = B  es lo mismo que  IF  A <> B
IF NOT X >= Y  es lo mismo que  IF  X < Y
etcétera
```

También existen relaciones lógicas entre valores que usan **AND**, **OR** y **NOT**; puede encontrarlas en el capítulo 10 del manual del ZX81. Debemos considerarlas como recursos que ahorran tiempo y memoria a los programadores expertos, pero no hacen nada que no pueda hacerse con las instrucciones que ya hemos estudiado en este libro.

Ahora ejercite usted su razonamiento lógico:

<u>Ejercicio 27.1</u> Otra vez el tanque de agua

El viejo sistema de tuberías estaba corroído y lo hemos desmontado. Ahora el tanque tiene una sola tubería de desagüe con tres grifos, A$, B$ y C$. Cambie las líneas **INPUT** que afectan a los grifos e introduzca esta nueva línea lógica:

```
15Ø IF A$ = "A" AND (B$ = "A" OR C$ = "A")
      THEN GOTO 1ØØØ
```

Ejecute el programa, abra y cierre los grifos y deduzca la nueva disposición de tubos y grifos.

En este capítulo hemos aprendido ...

Instrucciones lógicas **AND**, **OR** para ser utilizadas con **IF... THEN**.

AND tiene prioridad sobre **OR**.

Paréntesis para modificar o enfatizar prioridades.

28

Gráficos otra vez

Este capítulo trata de las figuras y gráficos en movimiento. En estos casos el ZX81 deberá funcionar en modo **SLOW**. Los usuarios del ZX80 tendrán que pasar al próximo capítulo.

Centelleos

Si deseamos remarcar determinadas palabras en la pantalla, podemos recurrir a la impresión inversa, a los caracteres intermitentes o a ambos, igual que en esta subrutina:

```
100  GOSUB 1000
900  STOP
1000 REM **RESPUESTA CORRECTA
1010 FOR J = 1 TO 20
1050 PRINT AT 15,20;"CORRECTO"  (impresión inversa)
1100 PRINT AT 15,20;"        "  (ocho espacios)
1200 NEXT J
```

Tal como está, el programa genera un centelleo muy rápido que conviene moderar. Esto puede hacerse insertando una instrucción **PAUSE** o bien, para un efecto más reposado, utilizando bucles vacíos:

```
1Ø6Ø FOR K = 1 TO 1Ø
1Ø7Ø NEXT K
111Ø FOR K = 1 TO 1Ø
112Ø NEXT K
```

Rebotes

Empezaremos dibujando fragmentos de suelo y de techo para que una pelota rebote en ellos:

```
1ØØ FOR J = 2Ø TO 4Ø
11Ø PLOT J,1
12Ø PLOT J,42
13Ø NEXT J
```

A continuación vamos a imprimir la pelota, cerca del techo:

```
1Ø  LET V = 1
2ØØ PRINT AT V,15;"O"
```

Y ahora hay que mover la pelota por la pantalla:

```
2Ø  LET VV = 1
15Ø LET V = V + VV
4ØØ GOTO 15Ø
```

Vemos una desagradable estela de bolas. Habrá que borrar las que vamos dejando atrás:

```
3ØØ PRINT AT V,15;" "
```

Ahora está un poco mejor, pero ¡parece que la pelota sea de plomo!. Para hacerla botar tenemos que invertir el signo de VV al llegar al suelo y al techo:

```
25Ø IF V = 2Ø OR V = 1 THEN LET VV = -VV
```

¡Perfecto!. Botará una y otra vez hasta que desconectemos o pulsemos **BREAK**.

Ahora vamos a extender el programa a dos dimensiones, con objeto de crear el fundamento de un juego de

televisión. La idea es la misma, pero ahora cambiaremos tanto el número de línea como el número de columna cada vez que se ejecute el bucle. Tendremos que limitar los rebotes al interior de un pequeño rectángulo para no agotar la memoria. La posición inicial de la pelota la estableceremos aleatoriamente.

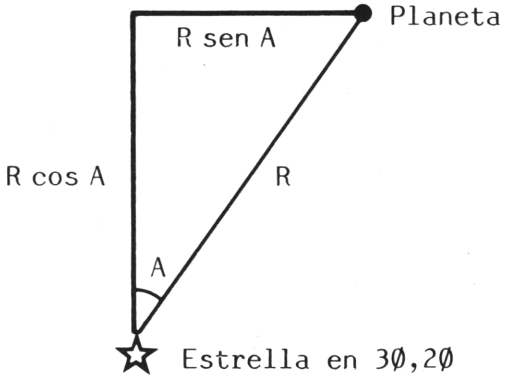
```
10  LET VV = 1
20  LET HH = 1
30  LET V = INT (RND * 13 + 1)
40  LET H = INT (RND * 19 + 1)
210 FOR J = 1 TO 42
220 PLOT J,42
230 PLOT J,13
240 NEXT J
250 FOR J = 14 TO 41
260 PLOT 1,J
270 PLOT 42,J
280 NEXT J
300 LET H = H + HH
310 LET V = V + VV
320 PRINT AT V,H;"O"
330 IF H = 20 OR H = 1 THEN LET HH = - HH
340 IF V = 14 OR V = 1 THEN LET VV = - VV
350 PRINT AT V,H;" "
360 GOTO 300
```

Satélites

Este programa sitúa una estrella en el centro de la pantalla y usa **PLOT** para hacer orbitar un planeta.

```
10  PRINT "¿RADIO?  DE 3 A 20"
20  INPUT R
30  PRINT AT 11,15;"*"
40  LET A = 0
100 UNPLOT 30 + R * SIN A,20 + R * COS A
110 LET A = A + .2
120 PLOT  30 + R * SIN A,20 + R * COS A
130 GOTO 100
```

El siguiente diagrama nos enseña un poco de trigonometría:



Si borra las líneas 1ØØ y 11Ø y añade éstas:

```
4Ø  FOR A = Ø TO 2*PI STEP .Ø5
13Ø NEXT A
```

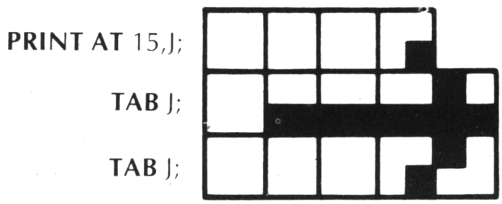
su programa dibujará un círculo.

Flechas

Aquí tiene un programa de tres líneas que lanza una flecha a través de la pantalla:

```
1ØØ FOR J = Ø TO 27
11Ø PRINT AT 15,J;"   ▗";TAB J;"▄▄▄▄▄▖"TAB J;"   ▗▘"
12Ø NEXT J
```

Los gráficos resultan difíciles de montar, pero un diagrama como éste nos ayudará:



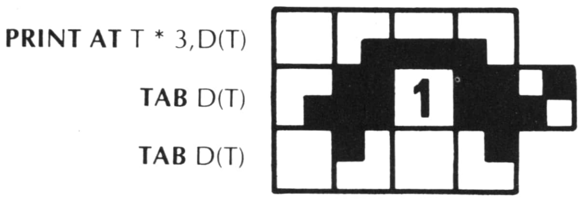
Aquí hay dos puntos importantes que hay que remarcar: (1) La técnica de empleo de **TAB** J para imprimir algo exactamente debajo de la misma posición de la línea anterior. (2) El uso de espacios al principio de las tres cadenas de caracteres para conseguir que la flecha vaya borrando automáticamente los restos de las anteriores a medida que avanza por la pantalla.

Tortugas rodantes

Aquí se combinan el programa sobre tiradas de dados del capítulo 25 y la técnica de lanzamiento de flechas anterior, para que cinco tortugas recorran la pantalla.

```
10  RAND
20  DIM D(5)
100 CLS
110 PRINT "CARRERA DE TORTUGAS ZX81"
200 LET T = INT (RND * 5 + 1)
210 LET D(T) = D(T) + 1
310 PRINT AT T * 3, D(T);"▄▄▄";TAB D(T);"▄█";T;
              " ██▛";TAB D(T);"▛▜"
320 IF D(T) < 27 THEN GOTO 200
400 PRINT AT 21,20;"NO. ";T;" GANA"
410 INPUT A$
420 RUN
```

El resultado es la imagen de una tortuga que se mueve por la pantalla igual que la flecha, aunque más lentamente. Este diagrama nos dará una idea más clara del gráfico:



Observe que las colas van dejando un rastro tras de sí a medida que las tortugas se desplazan. Si desea evitarlo tendrá que incluir un espacio en blanco justo detrás de la cola y acortar la carrera en un carácter.

Seguramente le apetecerá tratar de resolver algunos problemas de gráficos.

Ejercicio 28.1 Centelleo

Escriba una subrutina para premiar al ganador de alguno de sus juegos. Haga que la palabra 'GANADOR' centellee diez veces en la parte inferior derecha de la pantalla y quede fija al final.

Ejercicio 28.2 Pelota de goma

Hemos visto un programa sobre una pelota perfectamente elástica que bota indefinidamente. Ahora escriba un programa que simule una pelota más real, que bote verticalmente con botes cada vez más pequeños y que acabe quieta en el suelo. Es un problema difícil. Necesitará un bucle interno para hacer botar la pelota dentro de ciertos límites, y otro externo para ir reduciendo progresivamente el límite superior y, consecuentemente, la altura de los botes.

Ejercicio 28.3 Módulo lunar

Hemos hecho que flechas y tortugas se muevan por la pantalla. Su problema es ahora construir un pequeño módulo lunar (utilice los caracteres que prefiera) y hacer que descienda por la pantalla hasta la superficie de la luna. Quedará muy bien si consigue que vaya reduciendo su velocidad a medida que desciende.

29

¡Vaya memoria!

Aritmética binaria

Todos sabemos que los ordenadores trabajan en aritmética binaria (en base 2). Al igual que la mayoría de los microordenadores, el ZX81 contiene un gran número de células de memoria o bytes, cada uno de los cuales alberga un número de 8 bits. Aquí tiene una idea para construir un útil modelo de byte:

Corte una tarjeta por el eje mayor en dos mitades. Sobre una de ellas haga un dibujo como éste y practique cortes siguiendo las líneas a trazos:

1	1	1	1	1	1	1	1
128	64	32	16	8	4	2	1

Ahora doble las ocho pestañas de tal modo que los unos queden cubiertos, y escriba ceros sobre las caras que quedan a la vista, así:

0	0	0	0	0	0	0	0

Cada uno de los bits de su byte puede tener ahora el valor Ø (doblado) o 1 (desplegado). El valor decimal del número del byte se halla sumando los números decimales de las pestañas que estén desplegadas:

0	1	0	0	1	1	0	1
	64			8	4		1

En este caso, el número binario que figura en el byte es Ø1ØØ11Ø1, que es equivalente al número decimal 64 + 8 + 4 + 1 = 77.

Por supuesto, el número decimal más pequeño que podemos encontrar en un byte es Ø (todas las pestañas dobladas) y el mayor es 255 (todas desplegadas). Por tanto, las células de memoria del ZX81 están ocupadas por números comprendidos entre Ø y 255, y con ellos se pueden representar números, caracteres, instrucciones, etc. Los números mayores de 255 necesitan dos o más bytes, y cuando se define una variable como:

```
LET A =.1
```

el ZX81 toma cinco bytes para contener toda la posible información sobre A (tamaño, posición del punto decimal y signo) además de los que sean necesarios para el nombre de variable.

Memorias del ZX81

La memoria del ZX81 está dividida en dos partes. La ROM (read only memory) o memoria de solo lectura consiste en 8K bytes (1K = 2^{10} = 1024 bytes), que contienen todas las instrucciones permanentes necesarias para traducir el BASIC a código binario, y para indicar al ZX81 lo que tiene que hacer en cualquier caso. La ROM es permanente; usted puede averiguar lo que contiene un byte de la ROM, pero no puede cambiarlo en ningún caso.

A la otra parte la denominamos RAM (random access memory) o memoria de acceso aleatorio, y consiste en 1K byte (1Ø24 bytes numerados desde 16384 a 174Ø7). La RAM contiene toda la información que cambia de un programa a otro (las variables del sistema, su propio programa, el fichero de pantalla y las variables numéricas y de cadena). Usted puede averiguar el contenido de un byte de la RAM y también modificarlo.

Para aprovechar adecuadamente las posibilidades de su ZX81 necesitará la ampliación de RAM a 16K. Se trata de una caja del tamaño de un paquete de cigarrillos que encaja en un conector situado en la parte posterior del ZX81 y que amplía su memoria RAM hasta un total de 16K o 16383 bytes (vea el Apéndice 5).

¿Qué contiene ese byte?

Para saber lo que hay en un byte de ROM o de RAM cuya dirección es el número n, utilizaremos la función **PEEK** n. Aquí tiene un ejemplo:

```
1ØØ LET F = PEEK 16396 + 256 * PEEK 16397
11Ø PRINT "EMPIEZA VISION FICHERO","EN BYTE ";F
```

¿Pero qué está pasando?. Bien, la primera zona de RAM contiene las variables del sistema, es de tamaño fijo y se extiende desde 16384 hasta 165Ø9. La zona siguiente alberga su programa que, por supuesto, es de tamaño variable, seguido inmediatamente por el fichero de pantalla (la información que aparecerá en pantalla cuando el programa se detenga). Una de las variables del sistema es la dirección del principio del fichero de pantalla, que el ZX81 necesita conocer; es un número de cinco dígitos que ocupa dos bytes: el 16396 y el 16397.

Ejecute el programa, tome nota del principio del fichero de pantalla y luego añada esta línea:

```
12Ø PRINT
```

Si lo ejecuta de nuevo verá que el fichero de pantalla se ha desplazado seis bytes, que es el espacio

ocupado por la nueva línea de programa. Ahora vamos a averiguar lo que realmente hay en los primeros diez bytes del fichero de pantalla.

```
130 FOR J = 0 TO 9
140 PRINT PEEK F + J
150 NEXT J
```

Bueno, ya le advertí que los bytes de la ROM y de la RAM sólo contienen números hasta el 255. ¿Pero recuerda qué instrucción convierte los códigos en caracteres?¡Claro que sí, muy bien!.

```
140 PRINT CHR$ PEEK (F + J)
```

Ya dije que podíamos cambiar el contenido de cualquier byte RAM, aunque no es recomendable a menos que sepa perfectamente lo que está haciendo. La instrucción indicada es:

POKE m,n

siendo m la dirección del byte que estamos modificando y n el nuevo valor que introducimos (entre 0 y 255, por supuesto).

Pongamos un asterisco (código 23) en la línea superior de la pantalla:

```
125 POKE F + 5,23
```

Ejecute de nuevo el programa y asegúrese de que funciona bien. Se encuentra usted en el buen camino para averiguar cómo organiza su memoria el ZX81.

Programación avanzada

Se pueden escribir programas excelentes en BASIC sin recurrir a las instrucciones **PEEK** y **POKE**, pero finalmente se dará cuenta de que ellas nos permiten hacer cosas que de otro modo resultan imposibles. Usted también deseará emplear la función **USR** para escribir rutinas en código-máquina (se ejecutan con más rapidez

y ocupan menos memoria que en BASIC). Tendrá que leer cuidadosamente el manual del ZX81 (capítulos 25 a 28) y además comprar algún libro sobre programación avanzada. ¡Buena suerte!.

En este capítulo hemos aprendido ...

Instrucciones

POKE m,n para guardar el valor n en el byte de direccción m.

Funciones

PEEK m permite conocer el contenido del byte m en forma de número decimal.

Otros

La memoria del ZX81 de 8K ROM, 1K RAM y la expansión conectable a 16K RAM.

30

Puesta a punto

Si es capaz de escribir un programa de cierta longitud que funcione a la primera, es usted una persona que hace las cosas bien. Pero es más probable que al principio haya errores o 'bugs' que deba corregir.

Errores de sintaxis

Por lo general el ZX81 no le permitirá este tipo de equivocaciones. Olvídese una comilla o un paréntesis, mezcle variables numéricas y de cadena, o cometa alguna otra falta sintáctica y el ZX81 le mostrará el cursor [S] e impedirá la introducción de la línea. A propósito, asegúrese de que sus líneas son admitidas y evitará grandes pérdidas de tiempo tecleando al final líneas para sustituir a las erróneas.

Errores que detienen el programa

Aun en el caso de que todas las líneas hayan sido admitidas, puede suceder que se detenga la ejecución del programa a causa de algún otro error. En este caso el ZX81 nos ayuda imprimiendo un código que indica el número de línea y el tipo de error que ocasionó la parada. Estos códigos figuran en el Apéndice B del manual de operaciones del ZX81, y con frecuencia resul-

ta obvio lo que hay que hacer para poner las cosas en orden. Aquí tiene algunos casos en los que el remedio no es tan evidente.

Código 2/n. Variable no definida

Todas las variables tienen que definirse mediante una de las siguientes sentencias: **LET**, **INPUT**, **FOR** (para variables de control de bucles) o **DIM** (tablas).

Código 4/n. No queda memoria disponible

Es un contratiempo frecuente con la memoria de 1K RAM, que no permite ir muy lejos, sobre todo si se emplean gráficos y tablas. Estas son algunas ideas para economizar memoria. Recuerde que la RAM contiene el programa, las variables y el fichero de pantalla.

(1) Disminuya el número de variables. Reduzca las tablas y evite las variables ficticias innecesarias. Si le es posible, utilice el mismo nombre para más de una variable en diferentes partes del programa.

(2) Acorte las cadenas de caracteres y las variables de cadena; utilice abreviaturas.

(3) Elimine las líneas **REM**.

(4) Procure tener cuidado con las operaciones duplicadas. Póngalas en bucles o en subrutinas.

(5) Reduzca los gráficos y salidas por pantalla.

(6) Intente segmentar el programa. Recuerde que las variables definidas en una parte del mismo pueden utilizarse en otras.

(7) ¡Ahorre para comprar la ampliación a 16K RAM!.

Código 5/n. Pantalla completa

La instrucción **CONT** borra la pantalla y permite que el programa continúe. A la larga se verá obligado a buscar soluciones mejores, como reducir las salidas, o

bien intercalar pausas seguidas por **CLS**, o recurrir a la instrucción **SCROLL**.

Errores que no detienen el programa

A veces los programas parecen funcionar bien pero al final la salida no tiene ningún sentido. Recuerde el viejo proverbio que dice que no hay ordenadores malos, sino sólo programas deficientes. En ocasiones está claro que la salida no es correcta; en otras no es tan obvio y habrá que comprobarla cuidadosamente. He aquí algunas ideas al respecto:

(1) Verifique su programa introduciéndole datos que conduzcan a una salida conocida.

(2) Compruebe los resultados con una calculadora.

(3) Vigile los posibles errores de puntuación cuando tenga problemas con tablas de resultados o con gráficos.

(4) Pruebe las instrucciones condicionales introduciendo datos que cumplan la condición establecida y luego otros que no la satisfagan.

(5) Siga el curso de sus bucles (en especial los anidados) utilizando un diagrama de flujo.

(6) Ponga líneas **PRINT** provisionales en distintos puntos del programa para imprimir el valor de las variables.

(7) Interrumpa de vez en cuando la ejecución con líneas **STOP** provisionales y compruebe separadamente cada parte. Use un comando **PRINT** para ver las variables y luego **CONT** para seguir.

(8) Puede ser útil emplear **CLEAR**, como comando o como instrucción, para borrar las variables antes de introducir nuevos valores.

(9) Verifique las partes no iniciales del programa empleando **RUN** n o **GOTO** n para empezar la ejecución en la línea n. Recuerde que **RUN** borra todas las variables, y **GOTO** no.

Apéndice 1

BASIC ZX 81 en 8k ROM

Aquí se da una lista completa de las instrucciones BASIC accesibles desde el teclado del ZX81.

s — representa una cadena de caracteres entrecomillada o una variable de cadena.

n, m, p — simbolizan números, variables o expresiones. Cuando n, m y p deban ser enteros (como en **PLOT** n,m) el ZX81 redondea hasta el entero más próximo (por ejemplo, 10.4 se toma como 10, 10.6 como 11 y 10.5 como 11).

Comandos para escritura y edición de programas

EDIT — borra la parte baja de la pantalla y lleva allí la línea indicada por el cursor de programa para su edición.

⇧ ⇩ — mueven el cursor de programa una línea arriba o abajo.

⇦ ⇨ — mueven el cursor un carácter a la derecha o a la izquierda, sin afectar al texto.

FUNCTION — cambia el cursor a [F]. La siguiente tecla que se pulse llevará a la pantalla la correspondiente función y cambiará el cursor a [L].

GRAPHICS — cambia el cursor a [G], para obtener bloques gráficos e impresión inversa de letras, nú-

meros y otros caracteres de uso en cadenas. Pulse **GRAPHICS** otra vez para volver el cursor a la posición [L].

LIST Muestra todo el programa que quepa en pantalla empezando por la primera línea, y sitúa el cursor de programa frente a ella.

LIST n Muestra todo el programa que sea posible empezando por la línea n y pone frente a ella el cursor de programa.

NEWLINE (1) Transfiere una línea numerada y válida desde la parte inferior de la pantalla al programa.

(2) Hace que el ZX81 ejecute cualquier comando de la base de la pantalla.

(3) Limpia la pantalla después de haber ejecutado un programa y restablece en ella el listado del mismo.

RUBOUT Borra el carácter o palabra clave situados a la izquierda del cursor.

SHIFT Pulsando simultáneamente la tecla **SHIFT** y alguna otra, se llama al carácter impreso en rojo en esta última.

Comandos del sistema

Son instrucciones simbolizadas por una palabra clave que no forma parte del programa y que se ejecutan una sola vez tras teclearlas y pulsar **NEWLINE**. Aunque el ZX81 acepta cualquier palabra clave como comando, **INPUT** da un error de código Ø/8, y algunas otras carecen de sentido. Salvo **BREAK**, **STOP** y **COPY**, todos los comandos limpian la pantalla antes de ser ejecutados.

BREAK (1) Interrumpe al ZX81 mientras está trabajando. No se produce ninguna salida, excepto un código que informa de dónde se detuvo el programa.

(2) Para al ZX81 durante **LOAD** o **SAVE**.

CLEAR	Borra todas las variables.
CONT	Reinicia la ejecución del programa, detenida por **BREAK**, **STOP** o saturación de pantalla.
FAST	Pasa al ZX81 a modo **FAST** (4 x **SLOW**). En este modo, la pantalla permanece en blanco durante el proceso. Con el ZX80 éste es el único modo de trabajo posible.
GOTO n	Inicia la ejecución del programa en la línea n sin borrar ninguna variable.
LET	Define una variable.
LOAD s	Borra el contenido de la memoria del ZX81 y carga en ella un programa de nombre s, grabado en cinta magnética.
NEW	Borra el programa y las variables de la memoria del ZX81.
POKE m,n	Sitúa el valor n (de Ø a 255) en la dirección m de la memoria.
PRINT	Lleva a la pantalla todo lo que siga al comando **PRINT**.
RUN	Borra todas las variables y desencadena la ejecución del programa empezando por la primera línea.
RUN n	Borra todas las variables e inicia la ejecución, empezando en la línea n.
SAVE s	Envía el programa de nombre s desde la memoria del ZX81 a una cinta magnética para su almacenamiento permanente.
SLOW	Pasa al ZX81 del modo de trabajo **FAST** al modo **SLOW**, en el cual van apareciendo en pantalla todas las salidas a lo largo del proceso. Este es el modo en que se encuentra el ZX81 siempre que se le conecta; con el ZX80 no es posible.
STOP	Permite sacar al ZX81 de un bucle con **INPUT** cuando se le introduce como **INPUT**. Antes habrá que borrar las comillas de la cadena.

Sentencias de programa

Son palabras clave integradas como instrucción en el programa. El ZX81 acepta cualquier palabra clave, pero **CONT** y **NEW** rara vez tendrán sentido.

CLEAR	Borra todas las variables.
CLS	Limpia la pantalla.
DIM A(n)	Crea una tabla numérica de una dimensión, A(1), A(2),... A(n), y pone a cero todos sus elementos.
DIM A(n_1, n_2,... n_k)	Crea una tabla numérica de más de una dimensión y carga sus elementos con Ø.
DIM B$(n,m)	Crea una tabla de cadenas monodimensional cuyos elementos tienen un máximo de m caracteres, B$(1), B$(2),... B$(n) y les asigna una cadena de m espacios.
DIM B$($n_1$, n_2,... n_k,m)	Crea una tabla de cadenas multidimensional, con un máximo de m caracteres por miembro y cada uno de ellos cargado con una cadena de m espacios.
FAST	Pasa el ZX81 al modo **FAST** (vea el comando **FAST**).
FOR J = n **TO** m ... **NEXT** J	Crea un bucle **FOR... NEXT**. El valor inicial de J es n y va progresivamente aumentando en 1 tras cada ciclo. Cuando J > m el bucle ha finalizado y prosigue la ejecución del programa principal. Cada bucle se recorre n - m + 1 veces, o una sola si es m < n, y el valor final de J es igual a m + 1.
FOR J = n **TO** m **STEP** p	Modifica un bucle **FOR... NEXT** de tal manera que J es incrementado en p después de cada ciclo. Si se desea p puede ser negativo, resultando m < n.

GOSUB n	Salta a la subrutina que empieza en la línea n, continúa desde allí hasta encontrar un **RETURN** y entonces vuelve a la línea que sigue a **GOSUB** n.
GOTO n	Salta a la línea n del programa y continúa desde allí.
IF/THEN	Instrucción condicional. Si (**IF**) la condición se cumple, entonces (**THEN**) se ejecuta la sentencia (cualquier palabra clave). Si la condición no se cumple, el programa sigue a la próxima línea.
INPUT	Detiene el programa para que el operador asigne un valor a una variable numérica o de cadena.
LET	Asigna un valor a una variable numérica o de cadena.
PAUSE n	Detiene el programa y muestra una salida durante n/5Ø segundos, o bien hasta que se pulse alguna tecla. Si n > 32767 la pausa tiene una duración indefinida hasta que se oprima alguna tecla.
PLOT m,n	Ennegrece un pixel de pantalla, en la posición 'm horizontal, n vertical'. El rango es de m = Ø a 63 y n = Ø a 43, inclusive. La siguiente posición de **PRINT** se sitúa inmediatamente después de ese pixel.
POKE m,n	Guarda el valor n (Ø a 255) en la posición m de la memoria.
PRINT	Imprime cualquier cosa situada después de **PRINT** (números, variables numéricas o de cadena, expresiones y cadenas de caracteres) en la posición actual de **PRINT** en la pantalla.
PRINT AT m,n;	Imprime lo que esté después de **PRINT**, en la posición m líneas hacia abajo y n caracteres a la derecha, independientemente de la posición actual de **PRINT**.

PRINT TAB n;	Mueve la posición de **PRINT** hasta el carácter n de la línea actual (o de la siguiente si la posición actual de **PRINT** es mayor que n), e imprime allí cualquier cosa que siga a **PRINT**.
PRINT s (m **TO** n)	Imprime el fragmento de la cadena s que va desde el carácter m hasta el n. Si se omite m o n, se toma el primer o el último carácter.
RAND	Genera un número aleatorio como matriz para futuras expresiones **RND**.
REM	Precede a un comentario que será ignorado por el ZX81.
RETURN	Véase **GOSUB**.
RUN y **RUN** n	Borra todas las variables y reinicia el programa desde el principio, o bien desde la línea n.
SCROLL	Mueve el contenido de la pantalla una línea hacia arriba y sitúa la posición de **PRINT** al principio de la línea inferior.
STEP	Véase **FOR... NEXT... STEP**.
SLOW	Pasa al ZX81 a modo **SLOW** (vea el comando **SLOW**).
STOP	Para el program y las salidas. El comando **CONT** reanuda la ejecución.
UNPLOT m,n	Es exactamente igual que **PLOT**, excepto en que **UNPLOT** desennegrece el pixel.

Comandos e instrucciones para la impresora

COPY	Copia el contenido de la pantalla.
LLIST	Imprime un listado del programa.
LLIST n	Imprime un listado del programa, empezando en la línea n.
LPRINT	Imprime cualquier cosa que siga a **LPRINT**.

Funciones numéricas

ABS n	Valor absoluto de n (sin signo).
ARCCOS n	Angulo en radianes cuyo coseno es n.
ARCSIN n	Angulo en radianes cuyo seno es n.
ARCTAN n	Angulo en radianes cuya tangente es n.
COS n	Coseno de n (ángulo en radianes).
EXP n	Antilogaritmo neperiano de n: e^x.
INT n	Parte entera de n.
LN n	Logaritmo neperiano de n (en base e).
PEEK n	Valor que está ocupando la dirección n de la memoria.
PI (o π)	Número 3.14159...
RND	Número seudoaleatorio entre Ø y 1.
SGN n	Es el signo de n. Si n es positivo, entonces **SGN** n = 1; si n = Ø, **SGN** n = Ø, y si n es negativo, **SGN** n = - 1.
SIN n	Seno de n (ángulo en radianes).
SQR n	Raíz cuadrada de n.
TAN n	Tangente de n (ángulo en radianes).
USR n	Llama a la subrutina de código máquina en la dirección n.

Funciones de gestión de cadenas

CHR$ n	Carácter cuyo código es n (n puede estar comprendido entre Ø y 255 inclusive).
CODE s	Código numérico del primer carácter de s.
INKEY$	Lee todo el teclado. **INKEY$** es un solo carácter que corresponde a una tecla presionada, o a la cadena vacía si no se pulsa ninguna.
LEN s	Número de caracteres de la cadena s.

STR$ n	Convierte el número n en la aparentemente idéntica cadena 'n'.
VAL s	Siempre que sea posible, convierte la cadena s en un número, o en una expresión evaluable como un número.

Operadores lógicos

NOT **AND** **OR**	Se utilizan conjuntamente con IF en instrucciones condicionales.

Operadores aritméticos

n**m	Es n elevado a la m-sima potencia.
-n	Convierte en negativo el valor de n.
n*m	Es n veces m.
n/m	Equivale a n dividido por m.
n+m	Es n más m.
n-m	Es igual a n menos m.

Operadores relacionales

Se utilizan para comparar dos números, variables o expresiones. El signo = se usa también con **LET** para asignar un valor a una variable.

n = m	Es n igual a m.
n < m	Significa que n es menor que m.
n > m	Significa que n es mayor que m.
n <= m	Significa que n es menor o igual que m.
n >= m	Significa que n es mayor o igual que m.
n <> m	Significa que n es distinto de m.

Aquí también es válido utilizar **NOT**. Por ejemplo, **NOT** n = m es equivalente a n <> m.

Puntuación

; Instrucción utilizada para que la próxima salida **PRINT** se sitúe inmediatamente después de la anterior.

, Es una instrucción que se emplea para pasar a la siguiente zona **PRINT** y situar en ella la próxima salida. Cada línea de la pantalla se encuentra dividida en dos zonas **PRINT** iguales.

" Indica el principio o el final de una cadena de caracteres, de una cadena **INPUT**, o bien de la definición de una variable de cadena.

. Se utiliza como punto decimal.

"" Se usan para introducir comillas dentro de las cadenas de caracteres.

( ) Sirven para modificar las prioridades en las expresiones numéricas o en las instrucciones lógicas.

Con la excepción de ", todos los signos anteriores pueden usarse dentro de las cadenas de caracteres (y también : e ?).

Apéndice 2
Glosario

Aleatorio. Número: Número tomado de un conjunto cuyos miembros tienen todos la misma probabilidad de ser seleccionados y además dicha selección es independiente de cualquier acontecimiento previo.

Alto nivel. Lenguaje de: Lenguaje de programación formado por palabras inglesas fácilmente reconocibles.

Back-up: Cualquier método para el almacenamiento permanente de programas y variables. Por ejemplo, la grabación en cassettes.

Bajo nivel. Lenguaje de: Lenguaje de programación que utiliza el código máquina.

BASIC: Aunque originalmente fue concebido para principiantes, hoy es uno de los lenguajes de programación de alto nivel más ampliamente utilizados en microordenadores.

Binario. Número: Es un número en sistema binario (en base 2), en el que las únicas cifras son Ø y 1, en lugar de Ø, 1,... 9 del sistema decimal (base 1Ø).

Bit: Uno de los dos dígitos posibles de un número binario. Es decir, Ø o 1.

Bucle: Fragmento de un programa que se ejecuta repetidamente, a veces incluso miles de veces.

Bucles anidados: Son bucles situados dentro de otros

de tal manera que el bucle interno se ejecuta varias veces por cada vez que se recorre el externo que lo contiene.

Bug: Error en un programa suficiente para impedir que realice lo que se espera de él.

Byte: Un número binario de 8 bits de longitud. Es la unidad de almacenamiento más corriente de las memorias de los microordenadores.

Cadena de caracteres: Serie de caracteres entrecomillada e impresa literalmente por el ordenador.

Cadena vacía: Es una cadena que no contiene ningún carácter.

Carácter: Unidad de información susceptible de ser almacenada en un byte e impresa en pantalla. Por ejemplo A, 1, ; y **PRINT** son caracteres del ZX81.

Clave. Palabra: Es un comando, sentencia o función que ocupa un byte de memoria y que se introduce mediante una o dos pulsaciones sobre el teclado.

Código de carácter: Número de un solo byte que sirve para identificar un carácter. Puede variar de un ordenador a otro.

Código de error: Mensaje del ZX81 que aparece al final de una ejecución o tras una parada.

Código máquina: Código de programación que se caracteriza por utilizar el sistema hexadecimal para representar los números binarios.

Comando: Instrucción que no forma parte del programa y que hace que el ordenador cumpla determinada orden una sola vez.

Concatenacion: Unión de varias cadenas como si se tratara de eslabones.

Condicional. Instrucción: Sentencia que únicamente se ejecuta si se satisface determinada condición.

Diagrama de flujo: Representación gráfica de una serie de operaciones relacionadas entre sí y que deben realizarse siguiendo un orden determinado.

Dirección: Número que identifica un byte de memoria.

Editar: Seleccionar y modificar una línea determinada del programa.

Enter: Introducir en el ordenador a través del teclado una línea de programa, un comando o algún dato (en el ZX81 se hace presionando **NEWLINE**).

Entero: Número sin decimales que puede ser positivo o negativo.

Firmware: Se utiliza en ocasiones para referirse al intérprete y otros programas de la ROM.

Flowchart: Véase **diagrama de flujo.**

Función: Una operación determinada que hay que realizar con el número o la cadena que sigan.

Hardware: Partes físicas del ordenador y los dispositivos periféricos, en oposición a los programas.

Impresora: Dispositivo que conectado al ordenador permite obtener la salida impresa sobre papel.

K (de memoria): Unidad de memoria equivalente a 1Ø24 bytes.

Load: Cargar o transferir un programa desde el soporte del almacenamiento permanente al ordenador.

Notación científica: Expresa un número en términos de su mantisa (un número entre Ø y 1Ø) y su exponente (la potencia de 1Ø que hay que multiplicar por la mantisa para obtener el número original). El ZX81 utiliza este sistema con los números muy grandes o muy pequeños que de otra manera necesitarían más espacio del disponible en pantalla.

Pixel: Abreviatura de 'picture cell'. Es el elemento gráfico más pequeño que puede imprimirse en pantalla. En el caso del ZX81 la pantalla consta de 43 líneas de 63 pixels cada una.

Prioridad: Orden en el que se realizan las operaciones aritméticas o lógicas.

Programa: Sucesión numerada de instrucciones que tienen que ser ejecutadas por el ordenador.

RAM (random access memory): Memoria del ordenador que se utiliza para retener programas y datos. Cada byte de la RAM puede ser leído o alterado fácilmente a voluntad del programador.

Relacionales. Operadores: Símbolos como =, <, > que se usan para comparar números, expresiones o cadenas.

ROM (read only memory): Memoria permanente del ordenador que generalmente contiene el intérprete de BASIC, el sistema operativo y programas similares. Puede ser leída pero no modificada.

Save: Transferir un programa a un soporte de almacenamiento permanente para emplearlo en el futuro.

Sentencia: Es una instrucción para el ordenador que forma parte de un programa.

Seudoaleatorios. Números: Son números cuya distribución es aparentemente aleatoria, pero de hecho cada uno de ellos es calculado por el ordenador a partir de un número inicial y, por tanto, no son verdaderamente aleatorios.

Sintaxis. Error de: Error en la estructura de una línea del programa que impide que sea ejecutada y, en el caso del ZX81, incluso ser admitida.

Software: Manuales y programas del ordenador, en oposición a las partes físicas.

Subrutina: Fragmento de un programa al que puede ser dirigido el ordenador desde cualquier parte del programa principal. Una vez ejecutada la subrutina, el ordenador vuelve a la línea que siga inmediatamente al punto de partida.

Tabla de cadenas: Conjunto de variables de cadena que se identifican por el nombre de una tabla y unos subíndices. En el BASIC ZX81 una tabla de cadenas contiene una dimensión extra que indica la longitud de cada miembro.

Tabla numérica: Conjunto de variables numéricas, identificadas por un nombre de tabla y unos subíndices.

Variable de cadena: Es una variable que en BASIC se

identifica por un nombre seguido del signo $, y al que se pueden asignar cadenas de caracteres de cualquier tipo, con algunas excepciones irrelevantes.

Variable numérica: Variable representada por un nombre al que pueden asignarse cualquier número o expresión numérica.

Apéndice 3
Programas para el ZX 81

1. Rectángulos aleatorios (1K).
2. Espiral cuadrangular (1K).
3. Columnas aleatorias (1K).
4. Estadística de ventas (1K).
5. Media móvil (1K).
6. Múltiplos (1K).
7. Descomposición en factores (1K).
8. Cambio de base (1K).
9. Dibujos (1K).

9a. Dibujos y su almacenamiento en una tabla (16K).

10. Toros y vacas (1K).
11. Dado electrónico (1K).
12. Velocidad de reacción (1K).
13. Caja negra (16K).
14. Guía telefónica (16K).

1. Rectángulos aleatorios (1K)

Este programa utiliza parte de la pantalla (aproximadamente 2/3) para dibujar en ella una serie ilimitada de rectángulos de tamaño y en posición aleatorios.

```
10  RAND
100 LET A = INT(RND*43)
110 LET B = INT(RND*43)
120 LET C = INT(RND*43)
130 LET D = INT(RND*43)
140 IF A = C THEN LET A = A + 1
150 IF B = D THEN LET B = B + 1
200 FOR J = A TO C STEP SGN (C - A)
210 PLOT J,B
220 PLOT J,D
230 NEXT J
240 FOR J = B TO D STEP SGN (D - B)
250 PLOT A,J
260 PLOT C,J
270 NEXT J
300 GOTO 100
```

Lista de variables

A, B	Coordenadas de uno de los vértices de un rectángulo.
C, D	Coordenadas del vértice opuesto.
J	Variable de control de bucle.

Notas

Líneas 100 a 150	Establecen aleatoriamente las coordenadas de los vértices entre 0 y 43, y comprueban que A y C, y B y D no son iguales.
Líneas 200 a 230	Dibujan los lados horizontales. Dado que A puede ser mayor o menor que C, utilizamos **STEP SGN** (C - A), que puede adoptar los valores +1 o -1, para asegurar el buen funcionamiento del bucle **FOR/NEXT**.

Líneas 24Ø a 27Ø — Dibujan los lados verticales.

Línea 3ØØ — Retrocede para generar el próximo rectángulo. No necesitamos incluir un **RAND** en el bucle; en ciertos programas esto produciría incluso un efecto opuesto a la aleatorización deseada.

Con una RAM de 16K podemos ampliar el rango de A y C hasta 63, máximo permitido por **PLOT**. Este programa puede ser fácilmente modificado de manera que dibuje un número definido de rectángulos, o bien para utilizarlo como una subrutina que dibuje un rectángulo, dadas las coordenadas de dos vértices opuestos.

2. Espiral cuadrangular (1K)

Es un interesante, aunque poco útil, programa que dibuja y borra alternativamente una espiral cuadrangular en el centro de la pantalla. Usted tiene la posibilidad de modificarlo para que haga una espiral rectangular, útil para títulos.

```
10Ø LET S = 1ØØØ
```

```
1Ø  LET S = 1ØØØ
2Ø  LET D = 25
3Ø  LET H = 5
4Ø  LET V = 18
5Ø  LET S = 3ØØØ - S
9Ø  IF D = 1 THEN GOTO 2Ø
1ØØ FOR H = H TO H + D
11Ø GOSUB S
12Ø NEXT H
13Ø LET D = D - 1
2ØØ FOR V = V TO V + D
21Ø GOSUB S
22Ø NEXT V
23Ø LET D = D - 1
3ØØ FOR H = H TO H - D STEP -1
31Ø GOSUB S
32Ø NEXT H
33Ø LET D = D - 1
```

```
400  FOR V = V TO V - D STEP - 1
410  GOSUB S
420  NEXT V
430  LET D = D - 1
440  GOTO 90
1000 UNPLOT V,H
2010 RETURN
2000 PLOT V,H
2010 RETURN
```

Lista de variables

S	Determina qué subrutina entra.
D	Anchura de la espiral.
H, V	Coordenadas del punto de partida.

Notas

Línea 50	Asigna a S los valores 2000 y 1000 alternativamente para cada ejecución del bucle.
Línea 90	Detecta el final del bucle principal y entonces retrocede para fijar nuevos valores de las variables.
Líneas 100 a 120	Dibujan la primera línea vertical.
Línea 130	Reduce en 1 la longitud del lado.
Líneas 200 a 230	Dibujan la línea horizontal siguiente.
Líneas 230 a 430	Dibujan los dos lados que quedan.
Línea 440	Vuelve a la línea 90 para dibujar el elemento siguiente de la espiral.
Líneas 1000 a 2000	Son rutinas alternativas para dibujar o borrar la espiral.

3. Columnas aleatorias (1K)

Este programa imprime un grupo de 50 columnas de altura aleatoria y calcula e imprime la altura media de todas ellas.

```
5   LET T = 0
10  FOR J = 0 TO 49
20  LET R = INT(RND*40 + 1)
30  LET T = T + R
40  FOR K = 0 TO R
50  PLOT J,K
60  NEXT K
70  NEXT J
80  PAUSE 100
90  FOR J = 0 TO 49
100 PLOT J,T/50
110 NEXT J
120 PRINT TAB 5;"MEDIA R =";T/50
```

Lista de variables

T	Suma de los números aleatorios.
J,K	Variables de control de bucle.
R	Número aleatorio entre 1 y 40.

Notas

Líneas 10 a 30	Es una parte del bucle J que genera y totaliza 50 números aleatorios comprendidos entre 1 y 40.
Líneas 40 a 60	Bucle K que dibuja una columna de altura equivalente al valor actual de R.
Líneas 90 a 110	Traza una línea horizontal lo más próxima posible a la altura media de las 50 líneas verticales.
Línea 120	Se encarga de imprimir la media de los 50 números aleatorios.

4. Estadística de ventas (1K)

Se trata de un histograma de barras representativo de la venta de tuercas y tornillos durante los últimos cinco años.

```
10  DIM S(5,2)
100 FOR J = 1 TO 5
110 FOR K = 1 TO 2
120 LET S(J,K) = INT(RND*11 + 10)
130 NEXT K
140 NEXT J
200 PRINT "VENTAS EN CINCO AÑOS","DE TUERCAS ( ) Y
          TORNILLOS ( )",,,
250 PRINT "AÑO",,
300 FOR J = 1 TO 5
305 PRINT  1978 + J;" ";
310 FOR K = 1 TO 20
320 IF S(J,1) >= K AND S(J,2) >= K THEN PRINT "▓";
330 IF S(J,1) >= K AND S(J,2) < K THEN PRINT "▄";
340 IF S(J,2) >= K AND S(J,1) < K THEN PRINT "▒";
350 NEXT K
360 PRINT
370 PRINT
380 NEXT J
400 PRINT "    0 2 4 6 8 1 1 1 1 1 2" (4 espacios)
410 PRINT "              0 2 4 6 8 0" (14 espacios)
```

Lista de variables

S(5,2)	Tabla de 5 x 2 con las cifras de venta de 2 artículos durante 5 años.
J,K	Variables de control de bucle.

Notas

Líneas 100 a 140	Generan aleatoriamente un conjunto de cifras de venta comprendidas entre 10 y 20.
Líneas 300 y 360 a 380	Bucle exterior que tiene a su cargo la generación de cada uno de los cinco años.

Líneas 31Ø a 35Ø	Imprimen una barra en el gráfico correspondiente a un año e incluye un test para determinar cuál de los tres posibles bloques gráficos tiene que imprimirse.

5. Media móvil (1K)

La información de entrada consiste en una serie de datos que, por ejemplo, podrían ser cifras mensuales de ventas. El programa toma los N últimos datos (usted debe fijar el valor de N) y calcula la media y la desviación standard.

```
10  LET K = Ø
1ØØ PRINT "¿CUANTOS DATOS?"
11Ø INPUT N
12Ø DIM X(N)
2ØØ LET K = K + 1
21Ø PRINT "¿SIGUIENTE DATO?";
22Ø INPUT X(K)
23Ø PRINT X(K)
24Ø IF K < N THEN GOTO 2ØØ
25Ø CLS
3ØØ LET SX = Ø
31Ø LET SS = Ø
32Ø PRINT "ULTIMO ";N;" NUMERO",,
33Ø FOR J = 1 TO N
34Ø LET SX = SX + X(J)
35Ø LET SS = SS + X(J)**2
36Ø PRINT "    ";X(J)
37Ø IF J > 1 THEN LET X(J - 1) = X(J)
38Ø NEXT J
4ØØ PRINT ,,"MEDIA = ";SX/N
41Ø PRINT ,,"DESV. STD. = ";SQR(SS/N - (SX/N)**2),,,,
42Ø GOTO 21Ø
```

Lista de variables

K	Subíndice para el X(n) actual.
N	Número de datos a tomar cada vez.

X(N)	Tabla de N números.
SX	Suma de los últimos N números.
SS	Suma de los cuadrados de los últimos N números.

Notas

Líneas 1ØØ a 15Ø	Introduce el número de datos que deben ser promediados cada vez y las dimensiones de X(N) que resulten.
Líneas 2ØØ a 24Ø	Bucle **INPUT** para X(N). Al principio se introduce N veces y después sólo una vez para cada nuevo cálculo.
Líneas 33Ø a 38Ø	Bucle J que va tomando por orden cada uno de los X(N) números y realiza estas cuatro operaciones con ellos: 1. Los suma (SX). 2. Suma sus cuadrados (SS). 3. Los imprime. 4. Con la excepción de X(1), sitúa cada número en una posición de la tabla de tal manera que X(1) se pierde, X(2) se convierte en X(1), X(3) en X(2), y así sucesivamente.
Líneas 4ØØ a 41Ø	Calculan e imprimen la media y la desviación standard de los últimos N números.
Línea 42Ø	Retrocede para empezar con un nuevo X(N).

Evidentemente este programa puede simplificarse y servir para el cálculo de la desviación standard y la media de un conjunto dado de números.

6. Múltiplos (1K)

Aquí tenemos un programa que imprime en forma de cuadro los números comprendidos entre Ø y 1ØØ y utili-

za la impresión inversa para resaltar los múltiplos de cualquier número que se introduzca.

```
10  PRINT "TECLEE UN NUMERO ENTRE Ø Y 99"
2Ø  INPUT N
3Ø  CLS
1ØØ PRINT "LOS MULTIPLOS DE ";N;" SON"
11Ø IF N = Ø THEN LET N = 1ØØ
2ØØ FOR J = Ø TO 9
21Ø FOR K = Ø TO 9
22Ø IF J = Ø THEN PRINT " ";
23Ø LET M = 1Ø*J + K
24Ø IF INT (M/N)*N = M THEN GOTO 5ØØ
25Ø PRINT M;" ";
26Ø NEXT K
27Ø PRINT
28Ø PRINT
29Ø NEXT J
3ØØ GOTO 1Ø
5ØØ IF J > Ø THEN PRINT CHR$ (J + 156);
51Ø PRINT CHR$ (K + 156);" ";
52Ø GOTO 26Ø
```

<u>Lista de variables</u>

N	Número cuyos múltiplos se desea conocer.
J,K	Variables de control de bucle.
M	Número actual en el cuadro.

<u>Notas</u>

Línea 11Ø	Cambia N por 1ØØ cuando N = Ø para evitar un error en la línea 24Ø al dividir por cero. Obsérvese que Ø siempre aparece impreso en forma inversa, dado que es múltiplo de cualquier otro número.
Línea 23Ø	A partir de J y K genera el número actual en el cuadro.
Línea 24Ø	Verifica el número actual para ver si es múltiplo de N.

Línea 25Ø — Imprime normalmente los números que no son múltiplos.

Líneas 5ØØ a 51Ø — Imprime los múltiplos de N en forma inversa, partiendo del hecho de que el código de un número en impresión inversa es mayor en 156 que el propio número.

7. Descomposición en factores (1K)

La primera versión de este programa opera por iteración (repetición de las mismas operaciones una y otra vez). Toma un número dado y lo va dividiendo por 2 hasta que deja de ser divisible, luego empieza a dividir por 3, etc. Si el número no es divisible por ningún otro aparte de por él mismo, lo presenta como 'número primo'.

```
1ØØ PRINT "DESCOMPOSICION DE NUMEROS EN FACTORES"
11Ø PRINT "¿CUAL ES EL NUMERO?"
12Ø INPUT N
13Ø LET NN = N
14Ø LET F = 2
17Ø PRINT ,,,N;" = 1";
2ØØ IF N/F <> INT (N/F) THEN GOTO 3ØØ
22Ø PRINT " X ";F;
23Ø LET N = N/F
25Ø GOTO 2ØØ
3ØØ IF N = 1 THEN GOTO 4ØØ
33Ø LET F = F + 1
34Ø GOTO 2ØØ
4ØØ IF F = NN THEN PRINT "NUMERO PRIMO"
41Ø PRINT
42Ø PRINT,,"ESO ES TODO"
43Ø PRINT AT 21,19;"PULSE NEWLINE"
44Ø INPUT A$
45Ø CLS
46Ø GOTO 1ØØ
```

<u>Lista de variables</u>

N — Número a descomponer en factores primos.

NN	Variable ficticia.
F	Factor que se está probando.
A$	Cadena vacía que se introduce para continuar con otro número N.

Notas

Línea 13Ø	Asigna N a una variable ficticia NN.
Línea 14Ø	Selecciona el factor 2 para empezar.
Línea 2ØØ	Verifica si N es divisible por F.
Líneas 23Ø a 25Ø	N es divisible por F, entonces se divide N por F y se vuelve a la línea 2ØØ para seguir probando.
Línea 3ØØ	Comprueba si N ha sido reducido a 1, en cuyo caso se cesa de dividir por F.
Líneas 33Ø a 34Ø	N no es divisible por F, por lo que se incrementa F en 1 y se vuelve a la línea 2ØØ para probar otra vez.
Línea 4ØØ	Si F es igual a NN cuando todos los posibles factores han sido probados, entonces N tiene que ser un número primo.

El programa anterior trabaja bien pero es desesperantemente lento (véalo introduciendo 1998 y 1997). Esto sucede porque prueba como si fueran primos todo un conjunto de números que no lo son. Por ejemplo, se podrían descartar todos los números pares mayores que 2. O también, si tenemos un número primo, no tiene objeto tratar de dividir por ningún factor mayor que su raíz cuadrada. En consecuencia, vamos a teclear algunas líneas más que se deducen lógicamente de lo que acabamos de decir:

```
15Ø LET S = SQR N
16Ø LET PF = Ø
2ØØ IF N/F <> INT(N/F) OR NN = 2 THEN GOTO 3ØØ
24Ø LET PF = 1
3ØØ IF PF = Ø AND F > S OR N = 1 THEN GOTO 4ØØ
```

```
310 IF F = 2 THEN LET F = 1
330 LET F = F + 2
400 IF PF = 0 THEN PRINT " X ";NN;" NUMERO PRIMO"
```

Esto ya está mejor, pero aún puede mejorarse mucho más. Por ejemplo, pruebe introduciendo 3994, que es el doble de un número primo alto. Aprenderá mucho sobre programación y sobre números si intenta mejorar lo anterior utilizando un diagrama de flujo.

8. Cambio de base (1K)

Este programa convierte números de base 10 a base 2 o viceversa.

```
10  LET B$ = "BASE 10 NO. ="
20  LET C$ = "BASE 2 NO. ="
50  PRINT "CAMBIO DE BASE DE NUMERACION"
60  PRINT ,,"¿CAMBIO DESDE QUE BASE?","¿2 O 10?"
70  INPUT T
80  CLS
90  IF T = 2 THEN GOTO 600
110 PRINT
120 PRINT ,,B$;
130 INPUT T
140 PRINT T,,C$;
160 FOR J = INT(LN T/LN256*8) TO 0 STEP - 1
170 IF 2**J > T THEN GOTO 210
180 LET T = T - 2**J
190 PRINT "1";
200 GOTO 220
210 PRINT "0";
220 NEXT J
230 GOTO 110
600 LET T = 0
610 PRINT
620 PRINT ,,C$;
630 INPUT A$
640 PRINT A$
650 FOR J = 0 TO LEN A$ - 1
660 LET T = T + VAL A$ (LEN A$ - J)*2**J
670 NEXT J
```

```
680 PRINT B$;T
690 GOTO 600
```

Lista de variables

B$,C$	Cadenas que se utilizan varias veces.
T	1. Elección de la base de partida. 2. Número en base 10 a convertir. 3. Resultado de la conversión a base 2.
A$	Número en base 2 a convertir.

Notas

Línea 90	Bifurcación del programa en función de la base seleccionada.
Línea 160	Inicia el bucle J y fija el número de posiciones del número en base 2.
Línea 170	Verifica si el dígito actual debe ser 0 o 1.
Líneas 180 a 190	Si es 1, modifican la potencia de 2 en la expresión asignada a T e imprimen "1".
Línea 210	En otro caso, imprime "0".
Líneas 650 a 670	Toman los dígitos del número en base 2 de uno en uno, los multiplican por la potencia de 2 actual y los suman como T.

Hemos utilizado T como tres variables distintas para economizar memoria. Esto nos ha sido posible porque T se redefine en tres lugares diferentes. Naturalmente hay más procedimientos para convertir números a diferentes bases. Ensáyelos y pruebe con otras bases, especialmente con la hexadecimal que tiene singular importancia.

9. Dibujos (1K y 16K)

Este es un programa de 1K que le permitirá utilizar aproximadamente 2/3 de la pantalla para hacer dibujos

mediante un pixel gobernado por las cuatro flechas de las teclas 5, 6, 7 y 8. Si se pulsa D, el pixel va dejando un trazo continuo tras de sí. Si se presiona R, el pixel no deja ningún trazo y además borra todo lo que va encontrando a su paso.

```
10  LET X = 0
20  LET Y = 10
30  LET F$ = "R"
100 IF INKEY$ = "R" THEN LET F$ = INKEY$
110 IF INKEY$ = "D" THEN LET F$ = INKEY$
190 IF F$ = "R" THEN UNPLOT X,Y
200 IF INKEY$ = "5" THEN LET X = X - 1
210 IF INKEY$ = "6" THEN LET Y = Y - 1
220 IF INKEY$ = "7" THEN LET Y = Y + 1
230 IF INKEY$ = "8" THEN LET X = X + 1
300 IF X > 50 THEN LET X = 50
310 IF X < 0 THEN LET X = 0
320 IF Y > 43 THEN LET Y = 43
330 IF Y < 10 THEN LET Y = 10
340 PLOT X,Y
400 GOTO 100
```

Lista de variables

X,Y	Coordenadas del punto **PLOT/UNPLOT** actual.
F$	Indica si se está dibujando o borrando.

Notas

Líneas 100 a 400	Bucle principal que utiliza **INKEY$** para todas las entradas.
Líneas 100 a 110	Fijan la posición de F$ en "dibujo" (D) o en "borrado" (R).
Línea 190	**UNPLOT** que actúa solamente en modo "R".
Líneas 200 a 230	Controlan el movimiento del punto **PLOT** mediante las teclas correspondientes a las cuatro flechas de control del cursor.
Líneas 300 a 330	Mantienen la posición **PLOT** dentro de los límites del rectángulo definido.

En el programa para 16K el área de dibujo es algo menor, pero por lo demás la primera parte del programa trabaja igual que antes. Cuando usted haya completado su dibujo, pulse Z y todo el contenido del rectángulo será cargado por **PEEK** en una tabla. Y ahora, mediante **GOTO** 2ØØØ, puede reproducir el dibujo en la posición que desee de la pantalla. Si quiere puede guardar la tabla y las líneas 2ØØØ a 2Ø6Ø con objeto de utilizarlas más adelante.

Añada estas líneas al programa para 1K.

```
40   DIM A(80)
300  IF X > 19 THEN LET X = 19
330  IF Y < 28 THEN LET Y = 28
350  IF INKEY$ = "Z" THEN GOTO 1000
1000 LET F = PEEK 16396 + 256*PEEK 16397
1020 FOR J = 0 TO 7
1030 FOR K = 1 TO 10
1040 LET A(10*J + K) = PEEK (F + K + 33*J)
1050 NEXT K
1060 NEXT J
1070 STOP
2000 PRINT AT 5,10;
2010 FOR J = 0 TO 7
2020 FOR K = 1 TO 10
2030 PRINT CHR$ A(10*J + K)
2040 NEXT K
2050 PRINT TAB 10;
2060 NEXT J
```

Lista de variables

F	Byte inicial del fichero de pantalla.
J,K	Variables de control de bucle.
A(8Ø)	Conjunto de códigos numéricos que representan el contenido del área de dibujo.

Notas

Línea 35Ø	Orden para salir del bucle principal.

Línea 1ØØØ	Halla el principio del fichero de pantalla.
Líneas 1Ø2Ø a 1Ø6Ø	Determinan los códigos numéricos que correspondan al contenido del rectángulo de dibujo y los cargan en la tabla A(8Ø).
Líneas 2ØØØ y 2Ø5Ø	Establecen la nueva posición de impresión para el dibujo.
Líneas 2Ø1Ø a 2Ø6Ø	Reproducen el contenido del rectángulo de dibujo en la nueva posición.

10. Toros y vacas (1K)

Es una versión sencilla de un antiguo juego que ya empezamos en el Ejercicio 25.1. El ZX81 genera un número de cuatro cifras entre 1 y 6 (pueden estar repetidas) y usted dispone de nueve oportunidades para adivinarlo. Tras cada tentativa unas figuras negras indican el número de "toros" (cifra y posición correctas), y otras grises representan las "vacas" (cifras correctas en posición equivocada).

```
20  DIM N(4)
100 FOR J = 1 TO 4
110 LET N(J) = INT (RND*6 + 1)
120 NEXT J
200 FOR X = 1 TO 9
210 PRINT "ADIVINA EL NUMERO";X;"?";
220 INPUT G$
230 PRINT G$
300 FOR J = 1 TO 4
310 IF G$(J) = STR$ N(J) THEN GOSUB 1000
320 NEXT J
330 FOR J = 1 TO 4
340 FOR K = 1 TO 4
350 IF G$(K) = STR$ N(J) THEN GOSUB 1100
360 NEXT K
370 LET N(J) = ABS N(J)
380 NEXT J
390 PRINT
400 NEXT X
900 STOP
```

```
1ØØØ PRINT "█";
1Ø1Ø LET G$(J) = " "
1Ø2Ø GOTO 112Ø
11ØØ PRINT "▒";
111Ø LET G$(K) = " "
112Ø LET N(J) = -N(J)
113Ø RETURN
```

Lista de variables

N(4)	Cuatro cifras aleatorias entre 1 y 6.
X,J,K	Variables de control de bucle.
G$	Dato aventurado por el jugador en su intento de adivinar el número oculto.

Notas

Líneas 1ØØ a 12Ø	Generan un número de cuatro cifras que no se revela.
Líneas 3ØØ a 32Ø	Verifican los cuatro dígitos introducidos por el jugador para ver si son toros.
Líneas 33Ø a 38Ø	Comprueban esas mismas cifras para determinar las vacas.
Línea 37Ø	Restablece las cifras del número oculto para la próxima tentativa.
Líneas 1ØØØ a 113Ø	Subrutina que se ocupa del número de toros y de vacas. A ella se accede por diferentes puntos, según se trate de un toro o de una vaca. La salida es común.
Líneas 1Ø1Ø y 111Ø	Cancelan los dígitos introducidos que han resultado ser un toro o una vaca.
Línea 112Ø	Cancela un dígito del número oculto cuando ha sido adivinado, ya sea como toro o bien como vaca.

11. Dado electrónico (1K)

Este programa genera números seudoaleatorios del 1 al 6 y los convierte en imágenes de las caras de un dado corriente.

```
10   RAND
100  FOR J = 1 TO 9
120  PRINT AT J + 6,10;"█████████" (9 espacios inversos)
140  NEXT J
200  LET D = INT (RND*6 + 1)
210  GOSUB 1000 + D*100
220  INPUT A$
230  CLS
240  GOTO 100
1100 PRINT AT 11,14;" "         (1 espacio)
1110 IF D = 1 THEN RETURN
1200 PRINT AT 8,11;" "          (1 espacio)
1210 PRINT AT 14,17;" "         (1 espacio)
1220 RETURN
1300 GOTO 1100
1400 PRINT AT 8,11;" █████ "   (1 espacio, 5 espacios
                                 inversos, 1 espacio)
1410 PRINT AT 14,11;" █████ "  (igual que en 1400)
1420 RETURN
1500 PRINT AT 11,14;" "        (1 espacio)
1510 GOTO 1400
1600 PRINT AT 8,11;" ██ ██ "   (1 espacio, 2 espacios
                                 inversos, 1 espacio,
                                 2 espacios inversos,
                                 1 espacio)
1610 PRINT AT 14,11;" ██ ██ "  (igual que en 1600)
1620 RETURN
```

Lista de variables

J	Variable de control de bucle.
D	Número seudoaleatorio entre 1 y 6.
A$	Cadena vacía para volver a tirar el dado.

Notas

Líneas 1ØØ a 14Ø — Sirven para dibujar el cuadrado del dado.

Líneas 2ØØ a 21Ø — Generan un número aleatorio, D, y dirigen al ZX81 a la correspondiente subrutina.

Líneas 11ØØ a 162Ø — Son seis subrutinas para imprimir puntos sobre el dado. Forman un conjunto bastante complicado de instrucciones **GOSUB** y **RETURN** pero está concebido para minimizar la ocupación de RAM.

12. Velocidad de reacción (1K)

Si sigue las instrucciones, este programa medirá el tiempo que tarda usted en reaccionar y lo imprimirá en una escala de 1Ø a 6Ø. La precisión no es muy alta, pero los resultados son consistentes.

```
9Ø  PRINT AT Ø,Ø;"¿CUANTO TARDA EN REACCIONAR?    "
1ØØ PRINT " PULSE ALGUNA TECLA                 "
11Ø PRINT "   CUANDO LA PANTALLA SE BORRE      "
```

Utilice letras y espacios en impresión inversa.

```
16Ø PRINT AT 11,1;
17Ø FAST
19Ø PAUSE RND*3ØØ + 2ØØ
21Ø FOR A = Ø TO 62
22Ø IF INKEY$ <> " " THEN GOTO 47Ø
23Ø PRINT "▌";          (bloque gráfico SHIFT 5)
24Ø PRINT " ";          (cadena vacía)
25Ø NEXT A
46Ø IF A = 63 THEN PRINT "L E N T O"   (inverso)
47Ø IF A = Ø THEN PRINT "TRAMPA"       (inverso)
475 SLOW
49Ø PRINT
5ØØ FOR J = Ø TO 12
51Ø PRINT TAB J*5;J*5;
52Ø NEXT J
53Ø PRINT TAB 9;"MILISEGUNDOS"
54Ø PRINT,,TAB 5;"LA MEDIA ES 25"
```

```
550 PAUSE 200
560 CLS
570 GOTO 90
```

Lista de variables

A,J Variables de control de bucle.

Notas

Líneas 90 a 110 Se encargan de imprimir en negro el encabezamiento.

Líneas 170 a 190 Pasan a modo **FAST**. Luego, tras una pausa aleatoria, borran la pantalla.

Líneas 220 a 250 Bucle de control de tiempo cuya ejecución dura 1 milisegundo aproximadamente. La línea 220 provoca la salida del bucle cuando se pulsa alguna tecla.

Línea 470 Comprueba si se ha hecho trampa (presionar una tecla antes de que haya comenzado la ejecución del bucle de tiempo).

Líneas 475 a 540 Ocasionan la vuelta al modo **SLOW** e imprimen la escala.

13. Caja negra (16K)

Aquí tiene una versión para el ZX81 de un excelente juego de mesa, producido por Waddingtons y denominado "Caja negra". El tablero consiste en una cuadrícula de ocho por ocho con el perímetro numerado de 1 a 32. Hay cuatro átomos escondidos bajo la cuadrícula y usted debe hallarlos disparando rayos láser sobre la caja desde varias posiciones numeradas. Si alcanza un átomo el rayo es absorbido (este caso se representa por *). Si hay un átomo en una línea contigua a la del rayo, éste rebota en él y finalmente emerge adoptando la forma de letras que centellean. Si no hay átomos en las proximidades, el rayo va directamente a la caja. Tenga en cuenta que el rayo puede rebotar sobre más de

un átomo a su paso por la caja. Si el rayo se encuentra con átomos en las líneas de ambos lados, resulta reflejado hacia atrás (esto se representa por un pequeño cuadrado gris). También se produce una reflexión si hay un átomo en el borde del tablero junto al punto de entrada.

Se puede adivinar la posición de los átomos, pero ¡cuidado! hay una penalización de tres disparos por cada tentativa fallida. Si se rinde, el ZX81 le mostrará dónde estaban los átomos.

Resultaría muy pesado explicar todas las reglas del juego. No obstante usted las irá captando rápidamente. Este es un programa original. Existen otras versiones, pero me atrevo a suponer que mis gráficos son mejores que la mayoría de los otros.

```
5   RAND
1Ø  DIM A(1Ø,1Ø)
2Ø  LET S = 37
3Ø  LET B$ = "         "          (9 espacios)
4Ø  LET N$ = Ø
5Ø  LET RG = Ø
1ØØ FOR J = Ø TO 11
11Ø PRINT AT J + 5,9;" ████████████ " (12 espacios inversos)
12Ø NEXT J
13Ø PRINT AT 1Ø,1Ø;"CAJA NEGRA"   (letras inversas)
18Ø PRINT AT 21,2Ø;"ESPERE"
19Ø PAUSE 2ØØ
2ØØ FAST
2Ø5 CLS
21Ø GOSUB 1ØØØ
215 SLOW
22Ø GOSUB 12ØØ
23Ø GOSUB 33ØØ
25Ø PRINT AT Ø,22;"¿AHORA QUE?";TAB 23;" ";TAB 23;
""""S"" = DISPARO";TAB 23;" ";TAB 23;""""G"" = PRONOSTI-
CO";TAB 24;" ";TAB 24;" ";TAB 24;""""E"" = FIN"
27Ø GOSUB 42ØØ
28Ø GOSUB 33ØØ
285 IF I$ = "G" THEN GOTO 36ØØ
29Ø IF I$ = "S" THEN GOTO 3Ø5
295 IF I$ = "E" THEN GOTO 13ØØ
```

```
300  GOTO 250
305  LET NS = NS + 1
310  PRINT AT 10,25;"DISPARO";TAB 26;"NO.";NS
420  LET S = S + 1
430  LET S$ = CHR$ S
490  PRINT AT 0,21;"¿DESDE QUE";TAB 23;"NUMERO";
              TAB 24;"DISPARA";TAB 26;"USTED?"
500  INPUT N
510  IF N < 9 THEN GOSUB 1400
520  IF N > 8 AND N < 17 THEN GOSUB 2300
530  IF N > 16 AND N < 25 THEN GOSUB 2000
540  IF N > 24 AND N < 33 THEN GOSUB 1700
550  IF N > 32 THEN GOTO 500
560  GOTO 240
990  STOP
1000 REM**DIBUJO DE LA CAJA
1010 PRINT AT 3,0;
1020 FOR J = 1 TO 16
1030 PRINT "   ████████████████ "     (3 espacios, 16 inversos)
1040 NEXT J
1060 FOR J = 1 TO 8
1070 PRINT AT 0,2 + 2*J;J
1080 PRINT AT 1 + 2*J,21;J + 8
1090 PRINT AT 20,4;"2 2 2 2 2 1 1 1"
1095 PRINT AT 21,4;"4 3 2 1 0 9 8 7"
1100 PRINT AT 19 - 2*J,0;J + 24
1110 NEXT J
1120 FOR J = 6 TO 38 STEP 4
1130 FOR K = 6 TO 38
1140 UNPLOT J,K
1150 UNPLOT K,J
1160 NEXT K
1170 NEXT J
1180 RETURN
1200 REM**POSICIONAMIENTO DE CUATRO ATOMOS
1210 FOR J = 1 TO 4
1220 LET X = INT (RND*8 + 2)
1230 LET Y = INT (RND*8 + 2)
1240 IF A(X,Y) = 1 THEN GOTO 1220
1250 LETA(X,Y) = 1
1260 NEXT J
1270 RETURN
1300 REM**IMPRESION DE CUATRO ATOMOS
```

```
1310 FOR X = 2 TO 9
1320 FOR Y = 2 TO 9
1330 IF A(X,Y) = 1 THEN PRINT AT 21 - 2*Y,2*X;"*"
      (* en inverso)
1340 NEXT Y
1350 NEXT X
1360 GOTO 3900
1400 REM**MOVIMIENTO HACIA EL SUR
1410 LET X = N + 1
1420 LET Y = 10
1430 LET L = 2
1440 LET C = 2 + 2*(X - 1)
1450 PRINT AT L,C;S$
1460 FOR J = 1 TO 30
1470 NEXT J
1475 IF A(X,9) = 1 THEN GOTO 3000
1480 IF A(X - 1,Y - 1) = 1 OR A(X + 1,Y - 1) = 1
      THEN GOTO 2600
1490 IF A(X + 1,Y - 1) = 1 AND A(X - 1,Y - 1) = 1
      THEN GOTO 2600
1500 IF A(X + 1,Y - 1) = 1 THEN GOTO 2390
1510 IF A(X - 1,Y - 1) = 1 THEN GOTO 1790
1520 IF A(X,Y - 1) = 1 THEN GOTO 3000
1530 IF Y = 2 THEN GOTO 1560
1540 LET Y = Y - 1
1550 GOTO 1490
1560 LET L1 = 19
1570 LET C1 = 2 + 2*(X - 1)
1580 GOTO 4000
1700 REM**MOVIMIENTO HACIA EL ESTE
1710 LET X = 1
1720 LET Y = N - 23
1730 LET L = 19 - 2*(Y - 1)
1740 LET C = 2
1750 PRINT AT L,C;S$
1760 FOR J = 1 TO 30
1770 NEXT J
1775 IF A(2,Y) = 1 THEN GOTO 3000
1780 IF A(X + 1,Y + 1) = 1 OR A(X + 1,Y - 1) = 1
      THEN GOTO 2600
1790 IF A(X + 1,Y + 1) = 1 AND A(X + 1,Y - 1) = 1
      THEN GOTO 2600
1800 IF A(X + 1,Y - 1) = 1 THEN GOTO 2090
```

```
1810 IF A(X + 1,Y + 1) = 1 THEN GOTO 1490
1820 IF A(X + 1,Y) = 1 THEN GOTO 3000
1830 IF X = 9 THEN GOTO 1860
1840 LET X = X + 1
1850 GOTO 1790
1860 LET L1 = 19 - 2*(Y - 1)
1870 LET C1 = 19
1880 GOTO 4000
2000 REM**MOVIMIENTO HACIA EL NORTE
2010 LET X = 26 - N
2020 LET Y = 1
2030 LET L = 19
2040 LET C = 2 + 2*(X - 1)
2050 PRINT AT L,C;S$
2060 FOR J = 1 TO 30
2070 NEXT J
2075 IF A(X,2) = 1 THEN GOTO 3000
2080 IF A(X - 1,Y + 1) = 1 OR A(X + 1,Y + 1) = 1
        THEN GOTO 2600
2090 IF A(X + 1,Y + 1) = 1 AND A(X - 1,Y + 1) = 1
        THEN GOTO 2600
2100 IF A(X + 1,Y + 1) = 1 THEN GOTO 2390
2110 IF A(X - 1,Y + 1) = 1 THEN GOTO 1790
2120 IF A(X,Y + 1) = 1 THEN GOTO 3000
2130 IF Y = 9 THEN GOTO 2160
2140 LET Y = Y + 1
2150 GOTO 2090
2160 LET L1 = 2
2170 LET C1 = 2 + 2*(X - 1)
2180 GOTO 4000
2300 REM**MOVIMIENTO HACIA EL OESTE
2310 LET X = 10
2320 LET Y = 18 - N
2330 LET L = 19 - 2*(Y - 1)
2340 LET C = 19
2350 PRINT AT L,C;S$
2360 FOR J = 1 TO 30
2370 NEXT J
2375 IF A(9,Y) = 1 THEN GOTO 3000
2380 IF A(X - 1,Y + 1) = 1 OR A(X - 1,Y - 1) = 1
        THEN GOTO 2600
2390 IF A(X - 1,Y + 1) = 1 AND A(X - 1,Y - 1) = 1
        THEN GOTO 2600
```

```
24ØØ IF A(X - 1,Y + 1) = 1 THEN GOTO 149Ø
241Ø IF A(X - 1,Y - 1) = 1 THEN GOTO 2Ø9Ø
242Ø IF A(X - 1,Y) = 1 THEN GOTO 3ØØØ
243Ø IF X = 2 THEN GOTO 246Ø
244Ø LET X = X - 1
245Ø GOTO 239Ø
246Ø LET L1 = 19 - 2*(Y - 1)
247Ø LET C1 = 2
248Ø GOTO 4ØØØ
26ØØ REM**REFLEXION
26Ø5 FOR J = 1 TO 1Ø
261Ø PRINT AT L,C;" "      (1 espacio)
262Ø FOR K = 1 TO 2
263Ø NEXT K
264Ø PRINT AT L,C;"▒"      (1 GRAPHICS SHIFT A)
265Ø FOR K = 1 TO 3
266Ø NEXT K
267Ø NEXT J
268Ø RETURN
3ØØØ REM**ABSORCION
3Ø1Ø FOR J = 1 TO 1Ø
3Ø2Ø PRINT AT L,C;" "      (1 espacio)
3Ø3Ø FOR K = 1 TO 2
3Ø4Ø NEXT K
3Ø5Ø PRINT AT L,C;"*"
3Ø6Ø FOR K = 1 TO 3
3Ø7Ø NEXT K
3Ø8Ø NEXT J
3Ø9Ø RETURN
33ØØ REM**BORRADO PANTALLA SUPERIOR DERECHA
331Ø PRINT AT Ø,21;"  "    (2 espacios)
332Ø FOR J = Ø TO 21
333Ø PRINT AT J,23;B$
334Ø NEXT J
335Ø RETURN
36ØØ REM**ADIVINAR UN ATOMO
361Ø PRINT AT Ø,23;"¿DONDE ESTA";TAB 23;"EL ATOMO?"
362Ø PAUSE 2ØØ
363Ø PRINT AT 3,24;"¿CUADROS";TAB 24;"A LO LARGO?"
3635 GOSUB 42ØØ
364Ø LET X = VAL I$
3645 IF X > 8 THEN GOTO 3635
365Ø PRINT X
```

```
3660 PAUSE 50
3670 PRINT AT 6,24;"¿CUADROS";TAB 25;"ARRIBA?";
3675 GOSUB 4200
3680 LET Y = VAL I$
3685 IF Y > 8 THEN GOTO 3675
3690 PRINT Y
3700 IF A(X + 1,Y + 1) = 1 THEN GOTO 3800
3710 PRINT AT 21 - 2*(Y + 1),2*(X + 1);"0"  (inverso)
3720 PRINT AT 9,20;"AQUI NO HAY";TAB 26;"ATOMO"
3730 PAUSE 200
3740 PRINT AT  12,20;"PENALIZACION";TAB 22;"3 DISPAROS"
3750 LET NS = NS + 3
3760 PAUSE 200
3770 GOTO 240
3800 PRINT AT 21 - 2*(Y + 1),2*(X + 1);"*"  (inverso)
3810 PRINT AT 10,24;"MUY BIEN";TAB 24;"UNO"
3820 PAUSE 300
3830 LET RG = RG + 1
3840 IF RG = 4 THEN GOTO 4300
3850 GOTO 240
3900 REM**INDICACION DE SALIDA
3920 PRINT AT 20,24;"PULSE";TAB 24;"NEWLINE"
3925 INPUT C$
3930 CLS
3940 PRINT AT 5,0;"ESPERO QUE SE HAYA DIVERTIDO",,
          TAB 10;"VUELVA CUANDO QUIERA"
3950 STOP
4000 REM**CENTELLEO DE LETRAS
4010 FOR J = 1 TO 10
4020 PRINT AT L,C;" "      (1 espacio)
4030 PRINT AT L1,C1;" "    (1 espacio)
4060 PRINT AT L,C;S$
4070 PRINT AT L1,C1;S$
4080 FOR K = 1 TO 3
4090 NEXT K
4100 NEXT J
4110 RETURN
4200 REM**OBTENCION DE UN INKEY$
4210 IF INKEY$ <> " " THEN GOTO 4210
4220 IF INKEY$ = " " THEN GOTO 4220
4230 LET I$ = INKEY$
4240 RETURN
4300 REM**FELICITACIONES
```

```
4310 GOSUB 3300
4320 PRINT AT 0,24;"ADIVINO";TAB 24;"TODOS"
4330 PAUSE 100
4340 PRINT AT 3,24;"AL";TAB 24;NS;"º";
               TAB 24;"DISPARO"
4350 PAUSE 300
4360 PRINT AT 8,24;"¿JUEGA";TAB 22;"OTRA VEZ?;
               TAB 25;" ";TAB 25;"¿S/N?"
4370 INPUT C$
4380 IF C$ = "S" THEN RUN
4390 IF C$ = "N" THEN GOTO 3930
4400 GOTO 4370
```

Lista de variables

A(10,10)	Son los 64 cuadros de que consta la caja negra más una línea perimetral invisible.
B$	Cadena de 9 espacios.
NS	Número de disparo actual.
RG	Número de éxitos.
J,K	Variables de control de bucle.
I$	Valor actual de **INKEY$**
S$	Carácter que indica las posiciones de entrada y de salida del rayo.
S	Código del carácter anterior.
X,Y	Coordenadas.
L,C	Línea y columna del carácter que muestra la entrada del rayo.
L1,C1	Línea y columna del carácter que muestra la salida del rayo.
C$	Cadena input para proseguir con la ejecución del programa.

Lista de subrutinas

1000	Dibuja la cuadrícula y los números que la circundan.

1200	Sitúa cuatro átomos aleatoriamente.
1300	Imprime los cuatro átomos de la cuadrícula cuando el jugador se rinde.
1400	Se ocupa del movimiento de los rayos hacia el sur.
1700	Se encarga de los movimientos al este.
2000	Trata los movimientos hacia el norte.
2300	Realiza los movimientos al oeste.
2600	Imprime los caracteres que indican las reflexiones.
3000	Imprime los caracteres que simbolizan las absorciones.
3300	Borra la parte superior derecha de la pantalla.
3600	Pide al jugador que trate de adivinar la posición de un átomo y verifica si su pronóstico es correcto.
3900	Indica el fin de partida y se despide.
4000	Hace centellear los caracteres que señalan las posiciones de entrada y salida del rayo en la caja.
4200	Carga en **I$** el valor actual de **INKEY$**.
4300	Felicita al jugador que ha conseguido adivinar la posición de los cuatro átomos.

Notas

El "corazón" del programa está constituido por las cuatro subrutinas que se encargan de los movimientos y que son muy parecidas entre sí. Las notas siguientes se refieren a la subrutina que realiza los movimientos hacia el este:

Líneas 1710 a 1720	Fijan las coordenadas del punto de partida del rayo.

Líneas 173Ø a 174Ø	Fijan la posición del carácter que señala el punto de entrada del rayo.
Línea 1775	Verifica si existen condiciones para que se produzca una absorción en el borde.
Línea 178Ø	Comprueba si debe producirse una reflexión en el borde.
Línea 179Ø	Determina si tiene que existir una reflexión interna.
Línea 18ØØ	Verifica si hay un átomo en la línea de debajo de la de entrada, hecho que provocaría una reflexión hacia el norte.
Línea 181Ø	Comprueba si hay un átomo en la línea de encima de la de entrada, lo que ocasionaría una reflexión hacia el sur.
Línea 182Ø	Comprueba si hay un átomo en el cuadro siguiente, circunstancia que daría lugar a una absorción.
Línea 183Ø	Comprueba si se ha completado la trayectoria del rayo a través de la caja.
Líneas 184Ø a 185Ø	Si el rayo aún no ha salido, incrementan en 1 la coordenada X y hacen que se verifique todo de nuevo.
Líneas 186Ø a 188Ø	Fijan la posición del carácter que señala el punto de salida del rayo y remiten a la instrucción 4ØØØ para que centelleen los caracteres de entrada y salida.

14. Guía telefónica (16K)

Este es un ejemplo doméstico de una base de datos. Puede almacenar gran número de datos numerados en una tabla; en este caso se trata de nombres y números de teléfono de 2Ø caracteres como máximo. Este programa jamás debe ser ejecutado por medio del comando **RUN**, puesto que se perderían todos los datos que se hubieran introducido. Para ejecutarlo utilice siempre **GOTO 1Ø**. Entonces el ZX81 le ofrecerá la posibilidad de es-

coger entre listar los datos existentes, introducir un dato nuevo, encontrar uno concreto o borrarlo. En las tres últimas alternativas, el ZX81 emplea solamente los tres primeros caracteres del dato. De este modo, "busca Norman" conduciría también a Norton, Norden, North, etc.

Usted puede modificar el programa para que trate cualquier otro tipo de información que desee guardar. Puede cambiar el número de datos y su longitud. Con 16K RAM el límite es de unos 650 datos de 20 caracteres cada uno.

```
10  DIM N$(100,20)
20  LET N$(100) = "FIN"
30  LET E$ = "                    "          (20 espacios)
100 CLS
110 PRINT "GUIA TELEFONICA",,,,"¿QUE DESEA?",,,"LISTAR
          TODOS LOS NOMBRES = L","INTRODUCIR UN
          NOMBRE/NUMERO NUEVO = N","BUSCAR UN
          NUMERO = F",,"BORRAR UN NOMBRE/NUMERO = R"
120 INPUT Z$
130 IF Z$ = "N" THEN GOTO 500
140 IF Z$ = "F" THEN GOTO 200
150 IF ZS = "R" THEN GOTO 700
160 IF ZS = "L" THEN GOTO 400
170 GOTO 120
200 CLS
205 PRINT ,,,"¿NOMBRE POR FAVOR?"
210 INPUT Z$
215 IF LEN Z$ < 3 THEN GOTO 210
220 LET F = 0
230 LET X = 1
240 CLS
250 IF N$(X, TO 3) <> Z$(TO 3) THEN GOTO 300
260 LET F = 1
270 PRINT,,N$(X)
300 LET X = X + 1
310 IF N$(X, TO 3) <> "FIN" THEN GOTO 250
320 IF F = 0 THEN PRINT Z$;"NO ESTA"
330 GOTO 1000
400 CLS
410 LET X = 0
```

```
42Ø  LET X = X + 1
43Ø  SCROLL
44Ø  PRINT N$(X)
45Ø  IF N$(X, TO 3) <> "FIN" THEN GOTO 42Ø
46Ø  SCROLL
47Ø  GOTO 1ØØØ
5ØØ  CLS
51Ø  LET X = 1
52Ø  IF N$(X, TO 3) = "FIN" THEN GOTO 57Ø
53Ø  IF N$(X) = E$ THEN GOTO 6ØØ
54Ø  LET X = X + 1
55Ø  GOTO 52Ø
57Ø  PRINT,,"LO SIENTO - NO QUEDA ESPACIO"
58Ø  GOTO 1ØØØ
6ØØ  CLS
61Ø  PRINT "¿NOMBRE/NUMERO NUEVO?"
62Ø  INPUT N$(X)
63Ø  GOTO 1ØØ
7ØØ  CLS
71Ø  PRINT "¿QUE NOMBRE QUIERE BORRAR?"
72Ø  INPUT Z$
73Ø  LET X = 1
735  IF N$(X, TO 3) = "FIN" THEN GOTO 85Ø
74Ø  IF N$(X, TO 3) <> Z$(TO 3) THEN GOTO 9ØØ
745  CLS
75Ø  PRINT N$(X)
76Ø  PRINT "PRESIONE R PARA BORRAR",,,"O NEWLINE
            PARA EL SIGUIENTE ";Z$(TO 3)
77Ø  INPUT R$
78Ø  IF R$ <> "R" THEN GOTO 9ØØ
79Ø  PRINT,,N$(X);" BORRADO"
8ØØ  LET N$(X) = E$
81Ø  GOTO 1ØØØ
85Ø  CLS
86Ø  PRINT,,"NO HAY MAS ";Z$(TO 3);" NOMBRES EN
            LA GUIA"
87Ø  GOTO 1ØØØ
9ØØ  LET X = X + 1
91Ø  GOTO 735
1ØØØ PRINT AT 21,13;"PULSE N/L PARA MAS"
1Ø1Ø INPUT Z$
1Ø2Ø GOTO 1ØØ
```

Lista de variables

N$(1∅∅,2∅)	Tabla de 1∅∅ cadenas de 2∅ caracteres.
Z$,R$	Variables de cadena para **INPUT**.
F	Indicador de si un nombre se ha encontrado o no.
X	Subíndice actual.

Notas

Líneas 1∅∅ a 17∅	Imprimen un menú con cuatro alternativas y con una ramificación del programa en cuatro direcciones diferentes.
Líneas 2∅∅ a 33∅	Rutina de búsqueda de un nombre o número. Dado que sólo se consideran las tres primeras letras, pueden seleccionar más de un nombre a la vez.
Líneas 4∅∅ a 47∅	Hacen desfilar verticalmente la lista de los 1∅∅ nombres ordenados por subíndices.
Líneas 5∅∅ a 63∅	Buscan el primer miembro vacío de la tabla y entonces autorizan al usuario a cargar en él un nombre nuevo.
Líneas 7∅∅ a 91∅	Rutina para borrar un nombre. El usuario teclea el nombre que necesita borrar y el programa va presentando uno por uno todos los nombres cuyas tres primeras letras correspondan al nombre dado, con la opción de borrar o pasar al siguiente.

Apéndice 4

Ejemplos de soluciones de los ejercicios

Hay muchas maneras de escribir un programa. Tenga en cuenta que estas soluciones son meros ejemplos que utilizan exclusivamente las instrucciones que se han aprendido hasta el capítulo correspondiente. Las soluciones que usted proponga pueden ser otras e igualmente correctas.

Ejercicio 6.1 Cambio de línea

Teclee 15, **NEWLINE**, 2Ø, **NEWLINE** y luego:

```
1Ø PRINT "TRES LINEAS SE VAN, UNA SE QUEDA"
```

Ejercicio 6.2 Su dirección

Para borrar el programa anterior utilice **NEW**.

```
1ØØØ PRINT "SR. PEDRO GARCIA"
2ØØØ PRINT "ARCADIO BALAGUER, 13"
3ØØØ PRINT "CASTELLDEFELS"
4ØØØ PRINT "BARCELONA"
```

Ejercicio 7.1 Expresiones con paréntesis

(1) Fase 1 7 - 5 = 2   3Ø/12 = 2.5

```
    Fase 2           2*2.5 = 5
    Fase 3           5**3 = 125   (solución)

(2) Fase 1   6*8 = 48       23 - 11 = 12
    Fase 2   48 - 12 = 36   5 + 7 = 12
    Fase 3           36/12 = 3
    Fase 4           3**2 = 9     (solución)
```

Ejercicio 8.1 Cambio de moneda

```
10  LET R = 1.9
20  LET P = 75
30  LET D = 250
40  PRINT P*R
50  PRINT "DOLARES POR"
60  PRINT P
70  PRINT "£"
100 PRINT
110 PRINT
120 PRINT D/R
130 PRINT "£ SE NECESITAN PARA OBTENER "
140 PRINT D
150 PRINT "DOLARES"
```

Ejercicio 8.2 Paracaidismo

```
10 LET T = 10
20 LET A = 9.8
30 LET H = 3000 - A*T**2/2
40 PRINT "TIEMPO = "
50 PRINT T
60 PRINT
70 PRINT "ALTURA = "
80 PRINT H
```

La altura es de 2510 metros después de 10 segundos. Si introduce distintos tiempos en la línea 10 verá que tras 22 segundos de caída libre la altura es de 628 m. Es el momento de abrir el paracaídas.

Ejercicio 9.1 Círculos

```
10  LET R = 5
20  LET D = 2*R
30  LET C = 3.14*D
40  LET A = R**2*3.14
50  PRINT "    DATOS IMPORTANTES DE UN CIRCULO"
60  PRINT
70  PRINT "SI EL RADIO ES ";R;" CM"
80  PRINT
90  PRINT "DIAM = ";D;" CM","CIRCUNF = ";C;" SQCM"
100 PRINT TAB 8;"AREA = ";A;" SQCM"
```

Ejercicio 11.1 Parte decimal

```
10  LET N = 17.59
20  PRINT "NUMERO";TAB 10;"ENT";TAB 20;"DECIMAL"
30  PRINT
40  PRINT N;TAB 10;INT N;TAB 20;N - INT N
```

Ejercicio 11.2 Redondeo de decimales

```
10  LET N = 2.75
20  PRINT INT (N*10 + .5)/10
```

Ejercicio 12.1 Constructora

```
20  LET C = 500
30  LET Y = 1982
100 PRINT Y;" CAPITAL + INTERES = £";C
110 PRINT
120 LET Y = Y + 1
130 LET C = C*1.08
140 IF Y < 1990 THEN GOTO 100
```

Si desea redondear, cambie la línea 100 así:

```
100 PRINT Y;" CAPITAL + INTERES = £";INT (C*100
             + .5)/100
```

Ejercicio 12.2 ¿Cuáles son los años bisiestos?

```
1Ø  LET Y = 1982
1ØØ PRINT "AÑO"
11Ø PRINT Y;
12Ø IF Y/4 = INT (Y/4) THEN PRINT " AÑO BISIESTO";
13Ø LET Y = Y + 1
14Ø PRINT
15Ø IF Y < 2ØØØ THEN GOTO 11Ø
```

Ejercicio 14.1 Porcentajes

```
1Ø  PRINT "¿SU NOTA?";
2Ø  INPUT M
3Ø  PRINT M,,,"¿MAX NOTA POS?"
4Ø  INPUT MAX
5Ø  CLS
6Ø  PRINT M;" SOBRE ";MAX;" = ";M/MAX*1ØØ;
          " POR CIENTO",,,
7Ø  GOTO 1Ø
```

Ejercicio 14.2 Consumo de gasolina

```
1Ø  PRINT "¿CUANTOS KILOMETROS?"
2Ø  INPUT M
3Ø  PRINT M
4Ø  PRINT "¿LITROS CONSUMIDOS?"
5Ø  INPUT G
6Ø  CLS
7Ø  PRINT G;" LTS EN ";M;" KILOMETROS = ";M/G;
      " KM/LITRO"
8Ø  PRINT
9Ø  GOTO 1Ø
```

Ejercicio 16.1 Tabla de raíces cuadradas

```
1Ø  PRINT "NUMERO","RAIZ CUADRADA"
2Ø  PRINT
1ØØ FOR N = Ø TO 16
11Ø PRINT N,SQR N
12Ø NEXT N
```

Ejercicio 16.2 Múltiplos

```
10  PRINT "MULTIPLOS DE 4 HASTA 100"
20  FOR J = 0 TO 100 STEP 4
30  PRINT TAB 2*J;J;
40  NEXT J
```

Ejercicio 17.1 Tabla de multiplicar

```
10  FOR J = 1 TO 7
20  FOR K = 1 TO 7
30  PRINT TAB 4*K;J*K;
40  NEXT K
50  PRINT,,""
60  NEXT J
```

Ejercicio 17.2 Rectángulo

```
10  FOR J = 1 TO 5
20  FOR K = 1 TO 19
30  PRINT "■";
40  NEXT K
50  PRINT
60  NEXT J
```

Si quiere añadir un título, modifique la línea 10:

```
10  FOR J = 1 TO 4
```

y añada esta otra:

```
60  IF J = 2 THEN PRINT " ESTO ES UN RECTANGULO"
```

Utilice letras en impresión inversa para el título.

Ejercicio 18.1 Rellenar un formulario

```
10  PRINT "¿SU APELLIDO POR FAVOR?"
20  INPUT S$
30  PRINT,,"Y AHORA SU NOMBRE"
40  INPUT F$
50  PRINT,,"EDAD EN AÑOS POR FAVOR"
60  INPUT A$
```

```
70  PRINT,,"¿DONDE VIVE?"
80  INPUT T$
90  CLS
100 PRINT "MUCHAS GRACIAS ";F$;" ";S$
110 PRINT,,"USTED TIENE ";A$;" AÑOS"
120 PRINT "Y VIVE EN ";T$
```

Ejercicio 19.1 Elegir números

```
10  PRINT "TECLEE UN NUMERO ENTERO ENTRE 1 Y 99"
20  PRINT "    LUEGO PULSE NEWLINE"
30  INPUT N
40  IF N < 1 THEN GOTO 100
50  IF N > 99 THEN GOTO 100
60  IF N <> INT N THEN GOTO 200
70  GOTO 300
100 PRINT "NUMERO ENTRE 1 Y 99 POR FAVOR"
110 GOTO 30
200 PRINT "NUMEROS ENTEROS POR FAVOR"
210 GOTO 30
300 CLS
310 PRINT "EL NUMERO ES ";N
320 PRINT,,,,"SU CUADRADO ES ";N*N
330 PRINT,,,,,,"¿OTRO NUMERO?"
340 GOTO 30
```

Con los conocimientos que ha adquirido hasta ahora usted no puede evitar que el usuario introduzca letras (esto conduciría a un error de código 2/30). Para ello tendría que conocer la función **VAL** que se estudia más adelante.

Ejercicio 20.1 Ruleta

```
100 LET S = INT (RND*37)
110 IF S < 10 THEN PRINT " ";
120 PRINT S;"  ";              (2 espacios)
130 GOTO 100
```

Este programa incluye un cero. Pienso que esto es lo usual.

Ejercicio 20.2 Rectángulo aleatorio

```
100 LET N = INT (RND*15 + 1)
200 FOR J = 1 TO INT (RND*15 + 1)
210 FOR K = 1 TO N
220 PRINT "▨";          (un GRAPHICS SHIFT A)
230 NEXT K
240 PRINT
250 NEXT J
300 PAUSE 50
310 CLS
320 GOTO 100
```

Ejercicio 21.1 Volumen de un depósito

```
10   LET V = 0
110  PRINT,,,"¿DE QUE FORMA ES?"
120  PRINT,,"CILINDRICO - TECLEE CIL"
130  PRINT" O CUBICO - TECLEE CUBO"
140  INPUT A$
150  CLS
160  IF A$ = "CIL" THEN GOSUB 1000
170  IF A$ = "CUBO" THEN GOSUB 2000
180  IF V <> 0 THEN GOTO 300
190  PRINT "LA FORMA ";A$;" ES DESCONOCIDA"
200  GOTO 140
300  PRINT "VOL DE ";A$;" = ";V;" CM CUBICOS"
900  STOP
1000 REM** VOL DEL CILINDRO
1010 PRINT "¿ALTURA EN CM?";
1020 INPUT H
1030 PRINT H,,,"¿DIAMETRO EN CM?";
1040 INPUT D
1050 PRINT D
1060 LET V = PI*(D/2)**2*H
1070 RETURN
2000 REM**VOLUMEN DEL CUBO
2010 PRINT "¿LONGITUD DEL LADO EN CM?"
2020 INPUT E
2030 PRINT E
2040 LET V = E ** 3
2050 RETURN
```

En la línea 18Ø utilizamos V para crear un puente sobre las líneas 19Ø y 2ØØ en el caso de que el volumen ya haya sido calculado.

Ejercicio 22.1 Adivine un número

```
20  PRINT "¿CUAL ES MI CODIGO (1Ø A 99)?","DISPONE
            DE 8 OPORTUNIDADES",,,
1ØØ LET C = INT(RND*9Ø + 1Ø)
13Ø FOR J = 1 TO 8
14Ø PRINT "PRONOSTICO ";J;"?";
15Ø LET G$ = " "
2ØØ FOR K = 1 TO 2
21Ø IF INKEY$ <> " " THEN GOTO 21Ø
22Ø IF INKEY$ = " " THEN GOTO 22Ø
23Ø PRINT INKEY$;
24Ø LET G$ = G$ + INKEY$
25Ø NEXT K
31Ø LET G = VAL G$
32Ø IF G = C THEN GOTO 5ØØ
33Ø IF G > C THEN GOTO 37Ø
34Ø PRINT " DEMASIADO BAJO"
35Ø GOTO 4ØØ
37Ø PRINT " DEMASIADO ALTO"
4ØØ NEXT J
45Ø PRINT,,"ERA ";C
46Ø STOP
5ØØ PRINT " CORRECTO"
51Ø PRINT " ADIVINADO EN EL INTENTO NO ";J
```

En la línea 15Ø asignamos la cadena vacía a G$ para borrar pronósticos anteriores.

Ejercicio 23.1 Líneas verticales

```
1ØØ FOR K = Ø TO 43
11Ø PLOT Ø,K
12Ø PLOT 63,K
13Ø NEXT K
```

El programa no completará las verticales por insu-

ficiencia de memoria (estamos utilizando demasiado espacio en pantalla). Si reducimos la altura de 43 a 37 podremos obtener un rectángulo completo.

Ejercicio 23.2 Tarjeta de visita

```
10  FOR J = 12 TO 50
20  FOR K = 16 TO 30
30  PLOT J,K
40  NEXT K
50  NEXT J
100 PRINT AT 8,8;"JOHN JONES ESQ.,";TAB 9;"21
    OXFORD ROAD";TAB 10;"CHISWICK";TAB 12;"W.4."
    (todo en impresión inversa)
```

En la línea 100 se utiliza **PRINT AT** 8,8 para fijar la posición de impresión en la primera línea. Luego se emplea **TAB** para saltar a las otras líneas.

Ejercicio 23.3 Subrutina "continuamos"

```
100  PRINT "HAGAMOS UNA PAUSA"
110  GOSUB 1000
120  PRINT AT 5,0;"CONTINUEMOS"
900  STOP
1000 REM**CONTINUAMOS
1010 PRINT AT 21,19;"PULSE NEWLINE"
1020 INPUT A$
1030 PRINT AT 21,19;"             "    (13 espacios)
1040 RETURN
```

Ejercicio 24.1 CAN

```
10  LET L = 5
20  LET C = 0
30  PRINT "¿QUE ES UN CAN?"
40  PAUSE 300
100 PRINT AT 0,0;"TECLEE UNA PALABRA              "
           (32 posiciones entre letras y espacios)
110 INPUT W$
```

```
120 IF LEN W$ < 3 THEN GOTO 200
130 IF W$ (1 TO 3) = "CAN" THEN GOTO 300
140 IF W$ (LEN W$ - 2 TO LEN W$) = "CAN" THEN GOTO
        300
200 PRINT AT 0,0;W$;" NO ES CAN"
210 GOTO 40
300 PRINT AT 3,5;"LISTA DE CANS"
310 PRINT AT L,C;W$
320 LET L = L + SGN C
330 LET C = 15 - C
340 GOTO 100
```

Notas

La línea 12Ø rechaza las palabras de menos de tres caracteres.

Las líneas 13Ø y 14Ø aceptan todas aquellas palabras que empiecen o terminen por "can".

La línea 32Ø incrementa en 1 alternativamente el número que determina la línea de impresión.

La línea 33Ø fija la columna de impresión en Ø y 15 alternativamente.

Ejercicio 25.1 Toros y vacas sencillo

```
10  RAND
20  DIM N(4)
30  LET B = 0
100 FOR J = 1 TO 4
110 LET N(J) = INT (RND*6 + 1)
120 NEXT J
200 PRINT "ADIVINE EL NUMERO",,"CUATRO CIFRAS
          ENTRE 1 Y 6"
210 INPUT A$
220 CLS
230 PRINT "USTED HA DICHO ";A$
300 FOR J = 1 TO 4
310 IF N(J) = VAL A$(J) THEN LET B = B + 1
320 NEXT J
400 PRINT,,,"SU PUNTUACION ES DE ";B;" TOROS"
```

En el Apéndice 3 hemos incluido el programa de una versión más completa de este divertido juego. No obstante, es posible que prefiera intentar programarlo usted mismo.

Ejercicio 26.1 Calificación de un examen

```
10  DIM C$(6,6)
20  DIM M(6)
100 LET C$(1) = "SIMON"
110 LET C$(2) = "MARINA"
120 LET C$(3) = "GUILLERMO"
130 LET C$(4) = "EMILIO"
140 LET C$(5) = "JAIME"
150 LET C$(6) = "JUANA"
200 PRINT "¿NOMBRE DEL EXAMEN?";
210 INPUT T$
220 PRINT T$,"NOTA MAXIMA";
230 INPUT M
240 PRINT M
300 FOR J = 1 TO 6
310 PRINT C$(J),"¿NOTA?"
320 INPUT M(J)
330 PRINT M(J)
340 NEXT J
400 CLS
410 PRINT "EXAMEN DE ";T$,,,
420 PRINT "NOMBRE","PORCENTAJE",,,
430 FOR J = 1 TO 6
440 PRINT C$(J),M(J)*100/M
450 NEXT J
```

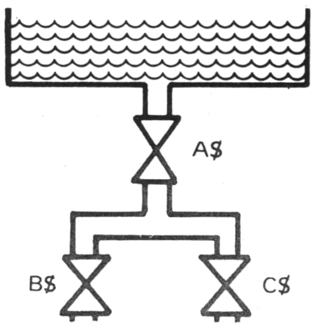
Con objeto de economizar memoria, en este programa hemos utilizado dos tablas monodimensionales paralelas, una para los nombres y la otra para las calificaciones. Con 16K y en un programa para una clase completa, podríamos usar una tabla de cadenas de dos dimensiones, introducir por **INPUT** las calificaciones como si fuesen cadenas y luego emplear **VAL** para transformarlas en números.

Ejercicio 26.2 Máquina tragaperras

```
1Ø  RAND
2Ø  DIM W$(6,7)
3Ø  DIM N(3)
1ØØ LET W$(1) = "CAMPANA"
11Ø LET W$(2) = "LIMON"
12Ø LET W$(3) = "CORONA"
13Ø LET W$(4) = "ANCLA"
14Ø LET W$(5) = "CEREZA"
15Ø LET W$(6) = "MANZANA"
2ØØ FOR J = 1 TO 3
21Ø LET N(J) = INT (RND*6 + 1)
22Ø PRINT AT 1Ø,(J - 1)*12;W$(N(J))
23Ø NEXT J
24Ø IF N(1) <> N(2) THEN GOTO 3ØØ
25Ø IF N(2) <> N(3) THEN GOTO 3ØØ
26Ø PRINT AT 18,15;"PREMIO"
27Ø STOP
3ØØ INPUT A$
31Ø GOTO 2ØØ
```

Aquí tenemos una tabla monodimensional de seis cadenas y otra de tres números aleatorios. En la línea 22Ø utilizamos uno de los elementos de la segunda tabla como subíndice de la variable de cadena de la otra. Las líneas 24Ø y 25Ø quedarían más elegantes si empleáramos un **AND** lógico, pero esto no se estudia hasta el próximo capítulo.

Ejercicio 27.1 Otra vez el tanque de agua



El agua podrá escaparse únicamente en el caso de que el grifo A$ permanezca abierto al mismo tiempo que B$ o C$.

Ejercicio 28.1 Centelleo

```
100  GOSUB 1000
900  STOP
1000 REM**CENTELLEO
1010 FOR J = 1 TO 10
1020 PRINT AT 21,15;"       "       (7 espacios)
1030 FOR K = 1 TO 10
1040 NEXT K
1050 PRINT AT 21,15;"GANADOR"      (letras inversas)
1060 FOR K = 1 TO 20
1070 NEXT K
1080 NEXT J
1090 RETURN
```

Ejercicio 28.2 Pelota de goma

```
10  LET V = 0
20  LET VV = 1
100 FOR J = 20 TO 40
110 PLOT J,1
120 NEXT J
140 FOR X = 0 TO 19
150 PRINT AT V,15;" "
160 LET  V = V + VV
170 PRINT AT V,15;"0"
180 IF V = X OR V = 20 THEN LET VV = - VV
190 IF V <> 20 THEN GOTO 150
200 NEXT X
```

Ejercicio 28.3 Módulo lunar

```
100 FOR L = 0 TO 18
110 PRINT AT L,15;" ";TAB 14;" A ";TAB
             14;"<S>";TAB 14;"| |"
120 FOR K = 0 TO L*2
```

```
130 NEXT K
140 NEXT L
```

La longitud de todas las cadenas que forman el módulo es de tres caracteres, y la S central debe estar en impresión inversa. El módulo es un plagio descarado del de Lunar Landing , un excelente juego de la serie producida en cassettes por Sinclair ZX Software.

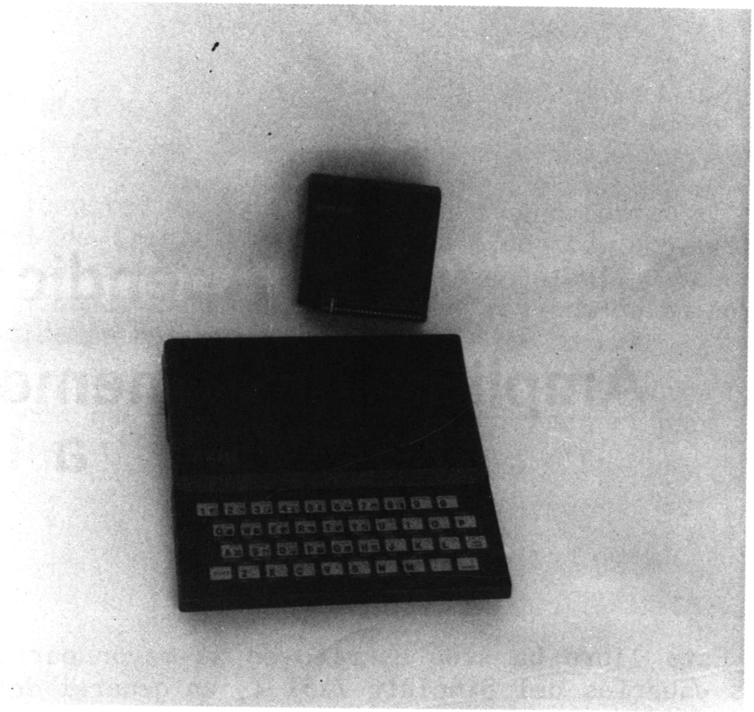
Las líneas 120 y 130 crean en el bucle principal una pausa que se va alargando uniformemente, para que la toma de tierra sea razonablemente suave.

Apéndice 5
Ampliación de memoria a 16K

Este libro ha sido escrito en su mayor parte para los usuarios del Sinclair ZX81 (y en general del ZX80 con BASIC en 8K ROM) con una memoria standard de 1K RAM a disposición del usuario para sus programas, datos, fichero de pantalla, etc. A medida que mejore su técnica de progrmación, verá que necesita disponer de una RAM mayor. Sinclair Research Ltd. suministra un módulo de expansión que se conecta en la ranura trasera del ZX81 o del ZX80 y que extiende la RAM hasta un total de 16K. Por un precio apenas superior a los dos tercios del del ZX81, constituye una excelente inversión en la situación actual del mercado.

El ZX81 ampliado permite abordar programas más largos (como el Programa 13 del Apéndice 3). También es útil para memorizar más datos, que pueden luego guardarse en cinta con el programa y usarlos más adelante (Programas 9 y 14 del Apéndice 3).

El empleo del módulo de memoria de 16K RAM no encierra dificultad alguna. Simplemente deberá tener la precaución de conectarlo antes de enchufar (nunca lo conecte o desconecte mientras el ZX81 permanece enchufado). Recuerde que los programas cortos ocupan más espacio en cinta cuando al grabarlos se está utilizando la ampliación de 16K. Es preferible conectar el módulo sólo cuando se desee cargarlos de nuevo.



Ampliación de memoria a 16K RAM conectable al ZX 81

INDICE ALFABETICO

* Títulos de los ejercicios incluidos en los capítulos 6 a 28
** Títulos de los programas del Apéndice 3